Stochastic approximation of a simple neural network-type learning algorithm via computer simulation

Israel Ncube[1], S. A. Campbell[2], and E. R. Vrscay[2]

Abstract *We present a computer algorithm for estimating the probability of convergence of a (random) neural network-type learning algorithm to a particular fixed point. Our algorithm is particularly useful for predicting the asymptotic behaviour of random algorithms which can converge to more than one fixed point. Our results suggest that the estimated probabilities are independent of the initialisation of the random learning algorithm.*

Key words. neural networks, learning, stochastic approximation

AMS subject classification. 92B20, 34K20, 34K15

1 Introduction

Consider a recursive random algorithm of the form

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \gamma_n h[\mathbf{x}_n, \varphi_n] \,, \tag{1}$$

where the $\mathbf{x}(n)$'s and $\varphi(n)$'s are in $\mathbb{R}^m$, $\{\gamma_n\}$ is a sequence of positive decreasing-to-zero real numbers such that $\sum_n \gamma_n = \infty$, $h : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is a continuous function, and $\varphi(n)$ is a sequence of random variables that are distributed according to some given law. The subject of stochastic approximation is concerned with characterisation of the long term behaviour of recursive random algorithms. Different approaches to this problem have been proposed [1-10].

A particular issue of interest may be posed by the following question: *Does the algorithm converge to a unique fixed point, independent of initialisation?* Ljung [1] and Kushner *et al.* [2] addressed this question by considering the ordinary differential equation (ODE)

$$\frac{d\mathbf{z}}{dt} = \overline{h}[\mathbf{z}(t)] \,, \tag{2}$$

where

$$\overline{h}(\mathbf{z}) = E[h(\mathbf{z},\, \varphi_n) \mid \varphi_0, \cdots, \varphi_{n-1}] \,,$$

[1]Dept of Mathematics and Statistics, York University, Toronto, Ontario, Canada M3J 1P3

[1]To whom correspondence should be addressed. Email: israel_n@mathstat.yorku.ca

[2]Dept of Applied Mathematics, University of Waterloo, Ontario, Canada N2L 3G1

for $\mathbf{z} \in \mathbb{R}^m$. Their famous theorem asserts that (1) converges, with probability one (wp1), to a stable equilibrium of (2) only if the latter has exactly one stable equilibrium point and no other stable structures.

However, if the ODE (2) has multiple such equilibria (with or without other stable structures), very little is known about the behaviour of (1). One of the few papers on this subject is that of Fort and Pagès [3]. They established and proved a theorem which permitted them to transfer the convergence of solutions of the associated ODE to that of the sequence of iterates generated by the random algorithm, in the case that the ODE has no *pseudocycles* (these include bona fide periodic orbits as well as isolated equilibria). Their approach is primarily a development of the original Kushner-Clark theorem. The proof of their theorem amounts to proving that the sequence of iterates generated by the algorithm has only one limiting point in the set of all equilibria of the associated ODE. Further, they also showed that if one of the elements of the above set is a saddle point, then the algorithm will not converge to it.

The main focus of this paper is the investigation, both qualitative and quantitative, of the convergence (or lack of thereof) of the sequence of iterates generated by a two-dimensional recursive random algorithm in the case that the associated ODE has two locally asymptotically stable equilibria and no other stable structures. This algorithm was first proposed by Ljung [1], and can be shown to be equivalent to the updating of weights in a neural network where competitive learning is used. The paper is organised as follows. In section 2, we introduce the classifier and define its task. Section 3 is a concise analysis of the associated system of ODE's, paying special attention to equilibria and how their numbers depend on some parameters in the model. In section 4, we outline the computer algorithm which is at the heart of the present paper. Section 5 is a discussion of simulation results. Section 6 is the closing discussion.

2 The classifier and its task

Consider a classifier which receives real-valued random signals, denoted as $\{\varphi_t\}_{t=1}^{\infty}$, belonging to two a priori unknown classes, A and B say, with probabilities $1-\lambda$ and λ, respectively, where $\lambda \in [0,1]$. Further, suppose that the signals are distributed according to the doubly-triangular probability density function (pdf) shown in Fig 1,

In [1], Ljung considered the case where the component triangles are not overlapping. Here we focus on the case when the two component triangles are overlapping, thus we assume that $\sigma_1 > 2 - \sigma_2$. The overlap interval is then $[1-\sigma_2,\ -1+\sigma_1]$. Note that our pdf is the superposition of the two component triangles. In addition, as is customarily done in the literature (see [2], for example), we assume that $\{\varphi_t\}_{t=1}^{\infty}$ is a sequence of independent, identically-distributed (with respect to the pdf shown in Fig 1) random variables.

The classifier is tasked with determining a real number $c(t)$ such that the signal φ_t is classified as

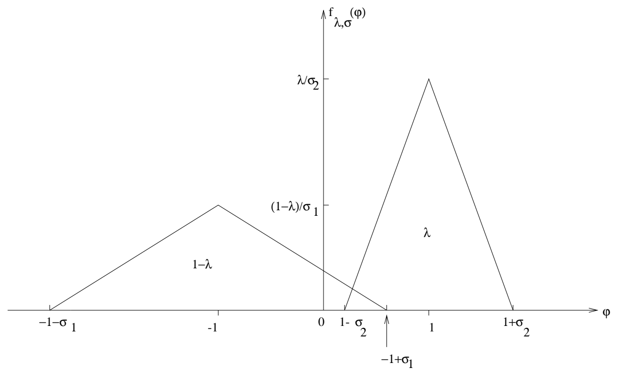

Figure 1: pdf of the signals to be classified by classifier. The parameters σ_1, σ_2 are in $\mathbb{R}$.

belonging to the class A if $\varphi_t \leq c(t-1)$, and as belonging to class B otherwise. To do this, we define variables $x_A(t)$ and $x_B(t)$ which represent estimates, at time t, of the mean value of signals classified as A or B, respectively. We then define the number $c(t)$ to be the bisector

$$c(t) \overset{def}{=} \frac{1}{2}[x_A(t) + x_B(t)] \ . \tag{3}$$

The variables $x_A(t)$ and $x_B(t)$ are updated according to the rules:

$$x_A(t) = x_A(t-1) + \begin{cases} \gamma_t[\varphi_t - x_A(t-1)] \ , & \text{if } \varphi_t \leq c(t-1) \\ 0 \ , \ \text{otherwise} \ , \end{cases} \tag{4}$$

$$x_B(t) = x_B(t-1) + \begin{cases} \gamma_t[\varphi_t - x_B(t-1)] \ , & \text{if } \varphi_t > c(t-1) \\ 0 \ , \ \text{otherwise} \ , \end{cases} \tag{5}$$

where $\{\gamma_t\}$ is a sequence of decreasing-to-zero positive real numbers such that $\sum_{t=1}^{\infty} \gamma_t = \infty$. Typically, γ_t is of the form

$$\gamma_t \overset{def}{=} t^{-p} \ , \ \text{for some } p \leq 1 \ . \tag{6}$$

The γ_t serve as 'training parameters' that modulate the correction terms. The requirement that $\gamma_t \to 0$ as $t \to \infty$ reflects the desire to gradually "phase out" the corrections in order to ensure convergence of the algorithm. The condition that $\sum_t \gamma_t = \infty$ is to ensure that the algorithm does not converge prematurely, i.e. that it converges to the "correct" point or set.

Consideration of the pdf in Fig 1 shows that it is desirable that

$$c(t) \to c_0 \in [1-\sigma_2, \ -1+\sigma_1] \text{ as } t \to \infty \ , \tag{7}$$

so that classification errors are minimised. In addition, it is clear that some of $\{\varphi_t\}_{t=1}^{\infty}$ will *always* be misclassified by $c(t)$, a direct result of the overlap of the two component triangles. As well, the resulting classification errors can be easily computed, as will be demonstrated in section 6. Now, by construction of our pdf, it can be seen that

$$-1-\sigma_1 < x_A(t) \leq x_B(t) < 1+\sigma_2 \ , \ \text{for all } t \ . \tag{8}$$

This defines an invariant region, denoted by $\overline{D}$, which is illustrated in Fig 2, in the (x_A, x_B) phase space.

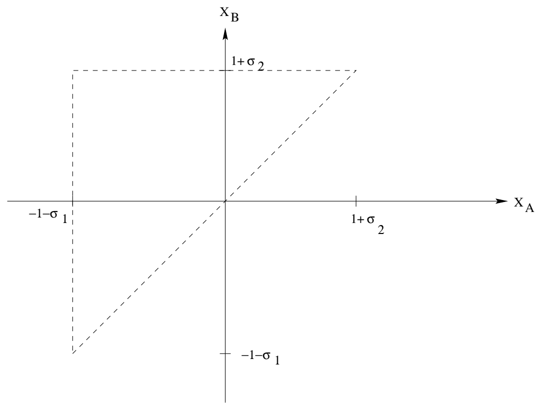

Figure 2: The invariant region $\overline{D}$ in (x_A, x_B) phase space.

3 The associated system of ODEs

The associated system of ODEs (9) is obtained by averaging, as in equation (2), with respect to φ_t the correction terms in (4)-(5). Consequently, we obtain

$$\begin{aligned} \dot{x}_A &= f_A(x_A, x_B, \lambda, \sigma_1, \sigma_2) \\ \dot{x}_B &= f_B(x_A, x_B, \lambda, \sigma_1, \sigma_2) \ , \end{aligned} \tag{9}$$

where f_A and f_B are defined by

$$\begin{aligned} f_A(x_A, x_B, \lambda, \sigma_1, \sigma_2) &\overset{def}{=} E_\varphi[\varphi - x_A] \ , \quad \varphi \leq c(t-1) \\ &= -x_A \int_{-1-\sigma_1}^{c} f_{\lambda,\sigma}(\varphi) d\varphi + \int_{-1-\sigma_1}^{c} \varphi f_{\lambda,\sigma}(\varphi) d\varphi \ , \end{aligned} \tag{10}$$

and

$$\begin{aligned} f_B(x_A, x_B, \lambda, \sigma_1, \sigma_2) &\overset{def}{=} E_\varphi[\varphi - x_B] \ , \quad \varphi > c(t-1) \\ &= -x_B \int_{c}^{1+\sigma_2} f_{\lambda,\sigma}(\varphi) d\varphi + \int_{c}^{1+\sigma_2} \varphi f_{\lambda,\sigma}(\varphi) d\varphi \ , \end{aligned} \tag{11}$$

where $c \overset{def}{=} \frac{1}{2}(x_A + x_B)$, and E_φ denotes the statistical expectation with respect to φ. The system of ODEs (9) has equilibria implicitly given by

$$(\bar{x}_A, \ \bar{x}_B) = \left(\frac{P(\bar{c})}{Q(\bar{c})} \ , \ \frac{E_\varphi - P(\bar{c})}{1 - Q(\bar{c})} \right) \ , \tag{12}$$

where $\bar{c} = \frac{1}{2}(\bar{x}_A + \bar{x}_B)$, and

$$
\begin{aligned}
P(c) &= \int_{-1-\sigma_1}^{c} \varphi f_{\lambda,\sigma}(\varphi) d\varphi \,, \\
Q(c) &= \int_{-1-\sigma_1}^{c} f_{\lambda,\sigma}(\varphi) d\varphi \,, \text{ and} \\
E_\varphi &= \int_{-1-\sigma_1}^{1+\sigma_2} \varphi f_{\lambda,\sigma}(\varphi) d\varphi \,.
\end{aligned}
\tag{13}
$$

Now, define the function

$$
\begin{aligned}
H_{\lambda,\sigma}(c) &= \frac{P(c)}{Q(c)} + \frac{E_\varphi - P(c)}{1 - Q(c)} - 2c \\
&= \frac{Q(c)[E_\varphi - P(c)] + [1 - Q(c)][P(c) - 2cQ(c)]}{Q(c)[1 - Q(c)]} \,,
\end{aligned}
\tag{14}
$$

where $P(c)$, $Q(c)$, and E_φ are as in (13). The $\overline{c}$ values needed to evaluate (12) are precisely the roots of $H_{\lambda,\sigma}(c)$. To compute these roots, we need only focus on the numerator

$$
F_{\lambda,\sigma}(c) \overset{def}{=} Q(c)[E_\varphi - P(c)] + [1 - Q(c)][P(c) - 2cQ(c)] \,. \tag{15}
$$

For illustrative purposes, we fix $\{\sigma_1,\ \sigma_2\} = \{1.0,\ 1.5\}$ and leave λ 'free'. Figs. 3-4 show plots of $F_{\lambda,\sigma}(c)$ vs c, for different values of λ.

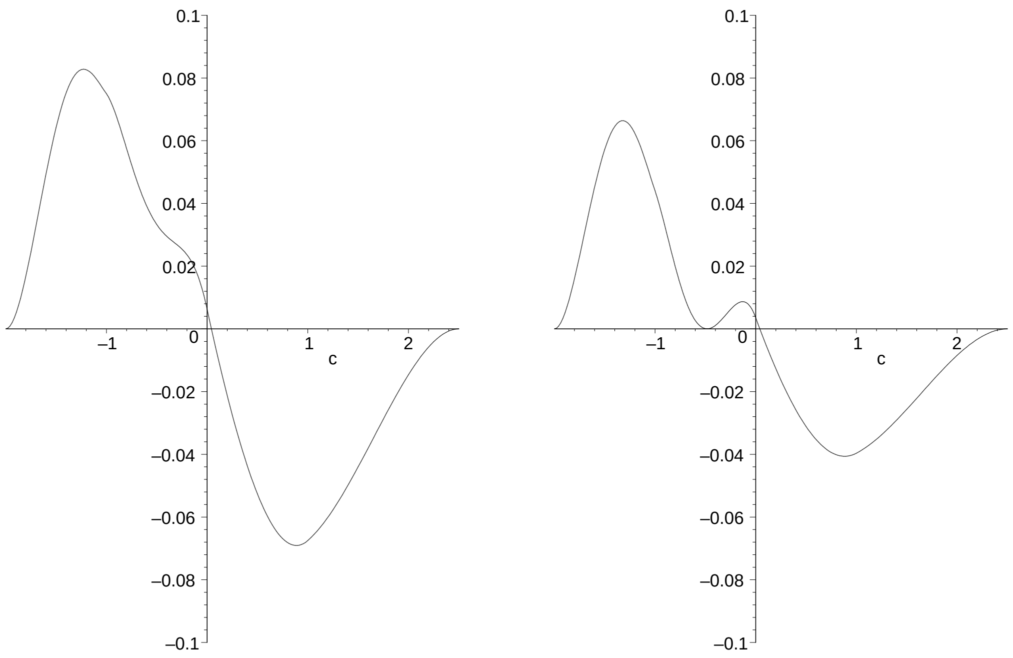

Figure 3: $F_{\lambda,\sigma}(c)$ with *left*: $\lambda = 0.1$, and *right*: $\lambda = 0.056$. In both cases, $\sigma_1 = 1.0$ and $\sigma_2 = 1.5$.

Observations:

For fixed $\{\sigma_1,\ \sigma_2\} = \{1.0,\ 1.5\}$, we have the following

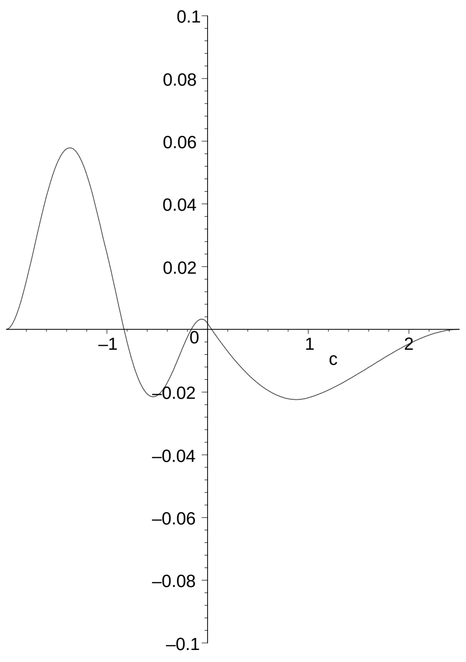

Figure 4: $F_{\lambda,\sigma}(c)$ with $\lambda = 0.03$, $\sigma_1 = 1.0$, and $\sigma_2 = 1.5$

- $\underline{\lambda > 0.056}$: The system of ODEs (9) has one stable equilibrium.
- $\underline{\lambda = 0.056}$: A Saddle-Node (S-N) bifurcation occurs, and a new root appears at $\overline{c} \approx -0.483$.
- $\underline{\lambda < 0.056}$: A S-N bifurcation has occurred, leading to the birth of two new equilibria of (9).
- $\underline{\lambda = 0.03}$: The function $F_{\lambda,\sigma}(c)$ has three distinct roots, viz. $\{\bar{c}_1, \bar{c}_2, \bar{c}_3\} \approx \{-0.830, -0.162, 0.040\}$.

It may be shown, using (12), that the three equilibria, arising from the case $\lambda = 0.03$, are given by

$$\begin{aligned}(x_{11}, x_{12}) &\approx (-1.2347, -0.4254) \quad \text{stable}\\ (x_{21}, x_{22}) &\approx (-1.0113, 0.6875) \quad \text{saddle pt}\\ (x_{31}, x_{32}) &\approx (-0.9983, 1.0791) \quad \text{stable}\,. \end{aligned} \tag{16}$$

In addition, it is clear that all the above equilibria lie inside the invariant region $\overline{D}$. For the case $\lambda = 0.03$, we pose the following question, which underpins the theme of the present paper: *Given any initial point in $\overline{D}$, to which of the two stable equilibria, (x_{11}, x_{12}) and (x_{31}, x_{32}), is the discrete system (4)-(5) 'likely' to converge?* We explore this question, with the aid of numerical simulations, in the next section.

4 Stochastic approximation via numerical simulation

In this section, we perform a numerical study to determine how an initial condition with unit probability temporally evolves, under the action of (4)-(5). This should tell us which stable equilibria of (9) the system (4)-(5) is likely to converge, in the case that (9) has multiple such equilibria. Recall that our phase space is the upper triangular region $\overline{D} : -1-\sigma_1 < x_A \leq x_B < 1+\sigma_2$. The computer algorithm employed to perform this investigation is outlined below.

- Divide $[-1-\sigma_1, 1+\sigma_2]^2$ into a grid of $npts \times npts$ cells, where, typically, $npts = 100$. Each cell is of size $del = \frac{2+\sigma_1+\sigma_2}{npts}$ and has a midpoint (x_A, x_B) which is considered to hold the mass of the cell.
- Initially, assign unit mass to a point $(x_A(0), x_B(0))$, and a mass of zero to the rest of the points. Store this "information" in a matrix P.
- Determine where the point (x_A, x_B) could go after a single iteration as outlined below. Note that one must know the current value of γ_t, the 'training' parameter. Now, where (x_A, x_B) goes depends upon the φ picked (according to our pdf). Since either x_A or x_B can be changed, one must examine all grid cells that lie in the same horizontal and vertical lines as (x_A, x_B).
- Suppose that we consider x_A as getting updated. Pick a grid cell and determine its left and right end points, x_1, x_2 say. Then the following issue arises:*For what values of φ does x_A get mapped to x_1, x_2?* Solve for these φ values, and denote them by φ_1, φ_2 respectively. The probability of choosing φ's lying in $[\varphi_1, \varphi_2]$ is

$$prob = \int_{\varphi_1}^{\varphi_2} f_{\lambda,\sigma}(\varphi) d\varphi \ , \tag{17}$$

  where $f_{\lambda,\sigma}(\varphi)$ is the pdf. Note that one must ascertain feasibility of the φ values. φ_1 and φ_2 may not lie in the ranges $[-1-\sigma_1,\ 1+\sigma_2]$ and $[-1-\sigma_1,\ c]$, the latter being responsible for updating x_A. This must be checked in each case, and appropriately remedied, for example using the following pseudo code:

```
        if $\varphi_1 \geq c$ go to 300
        if $\varphi_2 < -1-\sigma_1$ go to 200
        if $\varphi_2 \geq c$ set $\varphi_2 = c$
        if $\varphi_1 \leq -1-\sigma_1$ set $\varphi_1 = -1-\sigma_1$
200     continue
300     continue
```

  Also, one must check that the total sum of the probabilities of transferring to the various horizontal and vertical cells is unity.

- Store the updated mass in the matrix PP, which is initialised with zero entries. Iterate PP according to
$$PP[i,j] = PP[i,j] + prob \times P[k,l] \; ,$$
where i, j are the indices of the point being mapped to and k, l are the indices of the point being mapped from.

- Finally, set $P[i,j] = PP[i,j]$, and repeat the computations, recursively.

5 Simulation results

Some plots resulting from simulations of the above algorithm are shown in Figs. 5-8. Fig 5 (*left*) is a 3-dimensional "impulse" plot showing all those grid points $(x_A, x_B) \in \overline{D}$ at which the mass is greater than some arbitrary threshold, $\overline{p} = 0.001$ in this case, after 1000 iterations of the algorithm, with the unit mass initialised at $(-0.1, 0.1)$, and $\lambda = 0.1$. Further, in Figs. 5-6, note that the amount of separation of the mass "humps" is a numerical artefact, dependent on the value of the threshold $\overline{p} > 0$. The smaller the threshold, the bigger the amount of separation of the "humps", and vice-versa. The height of each "impulse" at each grid point gives the mass at that point. In theory, after N iterations of the algorithm, the total sum of masses at all the grid points in phase space $\overline{D}$ should be unity. In practice, however, because of the thresholding mentioned above, this sum will only be approximately unity.

In Fig 5 (*left*), it is clear that the entire mass migrates towards $(-0.9983, 1.0791)$. This behaviour is observed no matter where the algorithm is initialised inside $\overline{D}$. Recall that, for $\lambda = 0.1$, the associated system of ODEs (9) has exactly one globally asymptotically stable equilibrium at $(-0.9983, 1.0791)$. Thus, in accordance with the Kushner-Clark theorem [1], we expect the algorithm to converge, with probability one, to the equilibrium $(-0.9983, 1.0791)$. Fig 5 (*right*) depicts all those grid points (x_A, x_B) at which the mass is greater than the threshold $\overline{p} = 0.001$ after 1000 iterations of the algorithm, with the unit mass initialised at $(-0.1, 0.1)$, and for $\lambda = 0.03$. In this case, the associated system of ODEs (9) possesses two locally asymptotically stable equilibria, located at $(-0.9983, 1.0791)$ and $(-1.2347, -0.4254)$. It is clear that there is migration of a major portion of mass towards $(-1.2347, -0.4254)$ and a relatively minor one towards $(-0.9983, 1.0791)$. Fig 6 (*left*) shows the results of running the algorithm, starting with a unit mass positioned at $(-1.2347, -0.4254)$, with $\lambda = 0.03$. Fig 6 (*right*) depicts the results when a unit mass is initialised at $(-0.9983, 1.0791)$, with $\lambda = 0.03$.

Figs. 7 - 8 show projections, on the (x_A, x_B) plane, of Figs. 5 - 6. Here, each dot on the (x_A, x_B) plane denotes a grid point in $\overline{D}$ which has nonzero mass. In all the plots, we used $\gamma_n = (n+10)^{-0.25}$, which has a relatively slow rate of convergence. This is desirable in the sense that it minimises the chances of the algorithm getting trapped at some pseudo-equilibrium point [4]. Furthermore, we have only shown plots for fixed λ values. If we change the value of λ, the variation in the plots is minimal.

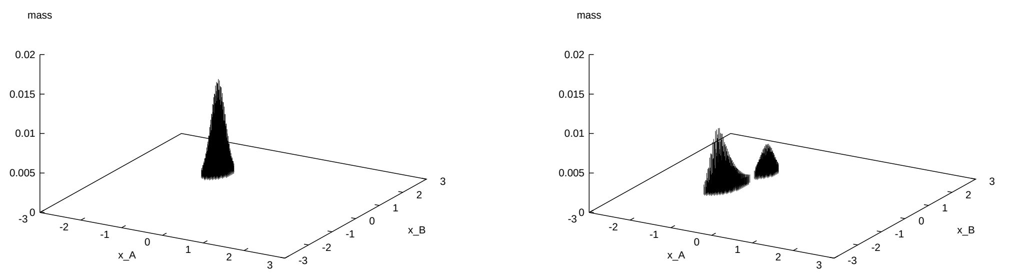

Figure 5: 3-dimensional "impulse" plot of all grid points whose associated masses are greater than $\overline{p} = 0.001$ after 1000 iterations of the algorithm, with *left*: $\lambda = 0.1$, and *right*: $\lambda = 0.03$. In both cases, the unit mass is initialised at the grid point $(-0.1, 0.1)$. The height of each "impulse" gives the mass at the grid point under consideration.

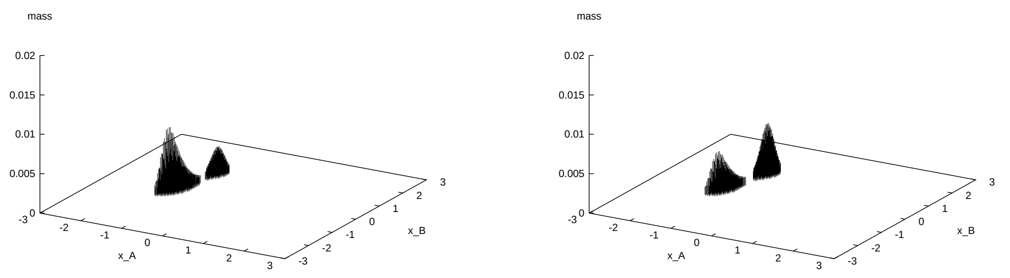

Figure 6: 3-dimensional "impulse" plot of all grid points whose associated masses are greater than $\overline{p} = 0.001$ after 1000 iterations of the algorithm, with *left*: $(x_A(0), x_B(0)) = (-1.2347, -0.4254)$, and *right*: $(x_A(0), x_B(0)) = (-0.9983, 1.0791)$. In both cases, $\lambda = 0.03$.

6 Discussion

We begin by noting that Fig 6 (*right*) is a transient. After more iterations, it does look like Figs 5 (*right*) and 6 (*left*). Simulation results suggest that the density of mass around the two equilibria (inside $\overline{D}$) of the associated system of ODEs (9) is almost independent of the initial condition of the algorithm. This mass density is indicative of the probability that the algorithm will converge to each respective equilibrium point, given some arbitrary initial point $(x_A(0), x_B(0)) \in \overline{D}$. Thus, the achievement of the algorithm is that, given some arbitrary initial point $(x_A(0), x_B(0)) \in \overline{D}$, we are able to say, in relative terms, what the probability of landing on $(-0.9983, 1.0791)$ or

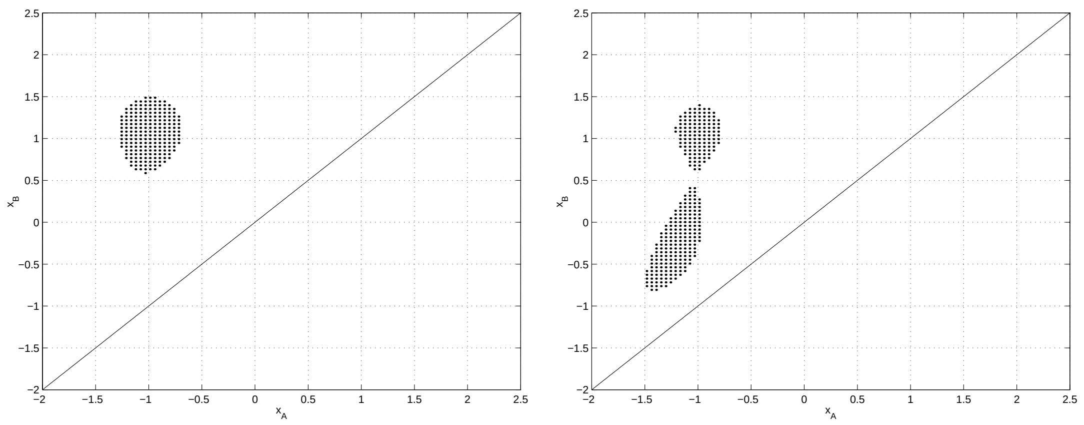

Figure 7: Grid points in $\overline{D}$ whose associated masses are greater than $\overline{p} = 0.001$ after 1000 iterations of the algorithm, with *left*: $\lambda = 0.1$, and *right*: $\lambda = 0.03$. In both cases, the unit mass is initialised at the grid point $(-0.1, 0.1)$.

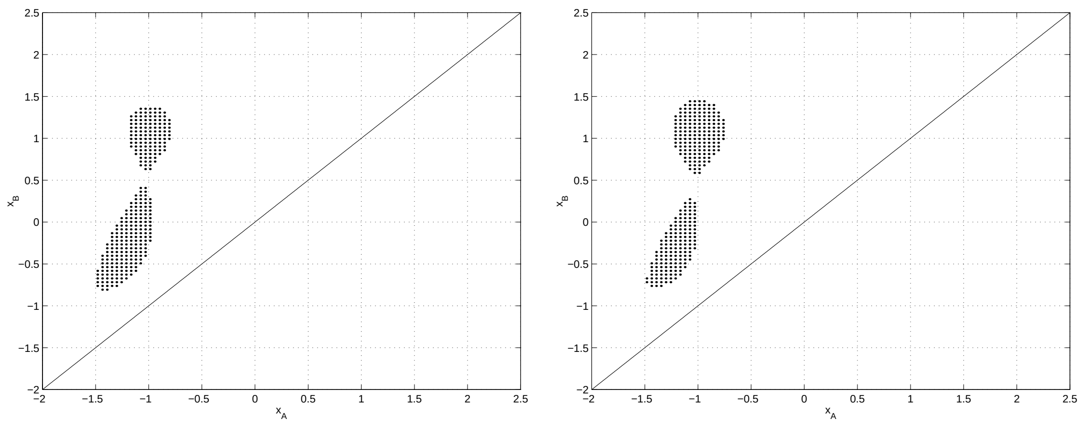

Figure 8: Grid points in $\overline{D}$ whose associated masses are greater than $\overline{p} = 0.001$ after 1000 iterations of the algorithm, with $\lambda = 0.03$. The unit mass is initialised at the point
left: $(x_A, x_B) = (-1.2347, -0.4254)$, and *right*: $(x_A, x_B) = (-0.9983, 1.0791)$.

$(-1.2347, -0.4254)$ is. The problem of numerically quantifying this probability is straightforward, since each dot represents a grid point whose associated mass is known.

Further, the numerics suggest that, for any initial point if $\overline{D}$, the algorithm is more likely to converge to the point $(-1.2347, -0.4254)$ than to $(-0.9983, 1.0791)$. Thus, our classifier (i.e. the number $c(t)$) will more likely converge to $\overline{c}_1 \approx -0.830$ than to $\overline{c}_3 \approx 0.040$, as $t \to \infty$. The classification rule $\overline{c}_1 \approx -0.830$ misclassifies 34.45% of the samples belonging to class A, while $\overline{c}_3 \approx 0.040$ misclassifies a mere 0.72% of samples belonging to class B. Finally, it is interesting to note that $\overline{c}_1, \overline{c}_3 \notin [1 - \sigma_2, -1 + \sigma_1] = [-0.5, 0]$, thus neither gives the optimal choice of c.

References

[1] Ljung L. *Analysis of Recursive Stochastic Algorithms* IEEE Transactions on automatic Control (1977), **AC-22**, 551-575.

[2] Kushner H.J. and Clark D.S. *Stochastic Approximation Methods for Constrained and Unconstrained Systems* New York (1978): Springer-Verlag.

[3] Fort J.C. and Pagès G., *Convergence of stochastic algorithms: From the Kushner-Clark theorem to the Lyapunov functional method* Adv. Appl. Prob. (1996), **28**, 1072-1094.

[4] Ncube I. *Stochastic Approximation of Artificial Neural Network-Type Learning Algorithms: A Dynamical Systems Approach.* Ph.D thesis, University of Waterloo, Canada (2001).

[5] Mackey C.M. and Lasota A. *Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics*, 2nd ed., Springer-Verlag (1994).

[6] Robbins H. and Monro S., *A stochastic approximation method* Ann. Math. Statist. (1951), **22**, 400-407.

[7] Fabian V. *On asymptotic normality in stochastic approximation* The Annals of Mathematical Statistics (1968), Vol. 39, No. 4, 1327-1332.

[8] Chung K.L. *On a stochastic approximation method* Ann. Math. Statist. (1954), **25**, 463-483.

[9] Kushner H. J., and Yin G. *Stochastic Approximation Algorithms and Applications.* Springer-Verlag, New York (1997).

[10] Benveniste A., Métivier M., and Priouret P. *Adaptive Algorithms and Stochastic Approximations.* Springer-Verlag (1987).