

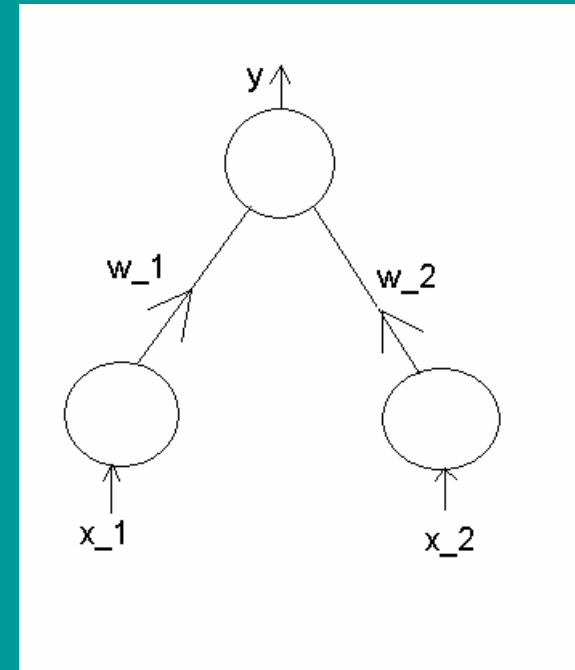
Learning Rate and attractor size of the single-layer perceptron

Martin S. Singleton, Alfred Hubler
Center for Complex Systems Research
University of Illinois

6th Understanding Complex
Systems Symposium

The Perceptron: Simplest Possible Neural Network

- As a dynamical system, a neural network has two types of dynamics (Hirsch, 1989):
 - Activation dynamics (how node variables evolve in time)
 - Weight dynamics (how weights connecting nodes evolve in time)
- Perceptrons have weight dynamics but no activation dynamics



rule $d^1 = (0, 0, 0, 1)$

"and rule"

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Recent Related Work

- Perceptrons can learn time series, generate chaotic time series (Freking et. al., 2002)
- Upper bounds for number of steps to converge for continuous perceptrons using online gradient learning (Wu et. al., 2003), and for boolean multilinear perceptrons (Carmesin, 1994)
- Learning generalization and information capacity of perceptrons studied via replica method (Watkin et. al., 1993)
- Order of magnitude comparisons of learning rates for batch, online, and cyclic learning (Heskes et. al., 1996)

Perceptron Algorithm

- Consider an N input perceptron at time step n, then the actual output when training rule k is:

$$y(w_n^k, x^j) = \Theta(w_n^k \cdot x^j + (-1)^{\lceil \frac{k}{2^{2^N-1}} \rceil} t)$$

$$j = n \bmod 2^N, \quad 0 \leq k \leq 2^{2^N} - 1$$

- The desired output d depends on the inputs, so the set of all such outputs specifies a rule (Boolean function), d^k which is a vector in $\{0,1\}^{2^N}$. Thus there are 2^{2^N} possible rules, only a small fraction of which are learnable (“linearly separable”)
- The inputs x are presented in cyclic, offline order. If the actual output doesn't equal the desired output, the weights are changed according to the prescription:

$$w_{n+1}^k = w_n^k + a(d_j^k - y(w_n^k, x^j))x^j$$

Equivalent Dynamics

- Specialize to case of two inputs (N=2)
- Antisymmetric rules: $d^k_2 \neq d^k_3$, e.g. $d^3=(0,0,1,1)$, $d^5=(0,1,0,1)$

Let k and \tilde{k} be antisymmetric rules, then if

$$\begin{pmatrix} w_{1,0}^k \\ w_{2,0}^k \end{pmatrix} = \begin{pmatrix} w_{2,0}^{\tilde{k}} \\ w_{1,0}^{\tilde{k}} \end{pmatrix}, \quad (1)$$

$$\begin{pmatrix} w_{1,4m}^k \\ w_{2,4m}^k \end{pmatrix} = \begin{pmatrix} w_{2,4m}^{\tilde{k}} \\ w_{1,4m}^{\tilde{k}} \end{pmatrix}, \quad m = 0, 1, \dots (2)$$

- Opposite rules: $\sim d^k = d^{-k}$, e.g. $d^1=(0,0,0,1)$, $d^{14}=(1,1,1,0)$

Let k and \tilde{k} be opposite rules, then if

$$\begin{pmatrix} w_{1,0}^k \\ w_{2,0}^k \end{pmatrix} = - \begin{pmatrix} w_{1,0}^{\tilde{k}} \\ w_{2,0}^{\tilde{k}} \end{pmatrix}, \quad (1)$$

$$\begin{pmatrix} w_{1,4m}^k \\ w_{2,4m}^k \end{pmatrix} = - \begin{pmatrix} w_{1,4m}^{\tilde{k}} \\ w_{2,4m}^{\tilde{k}} \end{pmatrix}, \quad m = 0, 1, \dots (2)$$

- 14 of 16 rules are linearly separable, by the above statements there are five classes (assuming isotropic initial weight distribution) of rules which display equivalent dynamics (in particular convergence rates identical)

Convergence Rates I (theory)

- If a rule k is learnable, the weight vector converges to a finite or infinite attractor A_k in \mathbb{R}^N , which is a cone formed by the intersection of two or more hyperplanes
- Given initial weight distribution with probability density function $p(x)$, the expected number of weight changes to converge is given by:

$$\mu_k = \int_{\mathbb{R}^N} p(x) \eta(A_k, x) dx$$

- There is a similar expression for the variance.
- The factor $\eta(A_k, w)$ is proportional to the length of the path from the initial weight vector w to A_k , and is found by solving:

$$w_{n+1} = w_n + a c_R,$$

where:

$$c_R = \frac{1}{2^N} \sum_{j=1}^{2^N} (d_j^k - y(\langle w^k \rangle_R, x^j)) x^j$$

Convergence Rates II (polynomial in the threshold t)

$$\mu_1(t; a, l) = \frac{22l^3 - 30l^2t + 30lt^2 - 49t^3}{48al^2} \quad (1)$$

$$\mu_2(t; a, l) = \frac{30l^3 - 30l^2t + 27lt^2 - 11t^3}{24al^2} \quad (2)$$

$$\mu_3(t; a, l) = \frac{103l^3 + 93l^2t + 97lt^2 + 7t^3}{192al^2} \quad (3)$$

$$\mu_4(t; a, l) = \frac{113l^3 + 3l^2t + 93lt^2 - 7t^3}{192al^2} \quad (4)$$

$$\mu_5(t; a, l) = \frac{44l^3 + 84l^2t + 42lt^2 - t^3}{96al^2} \quad (5)$$

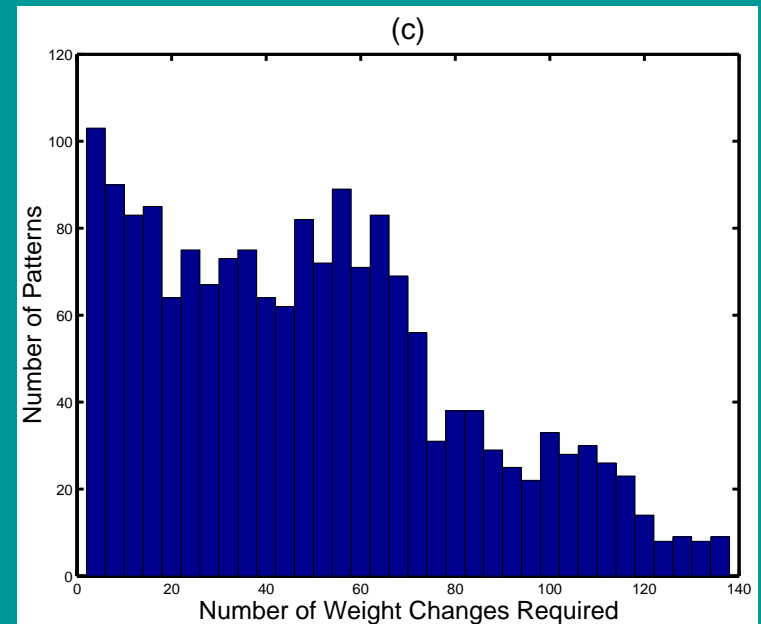
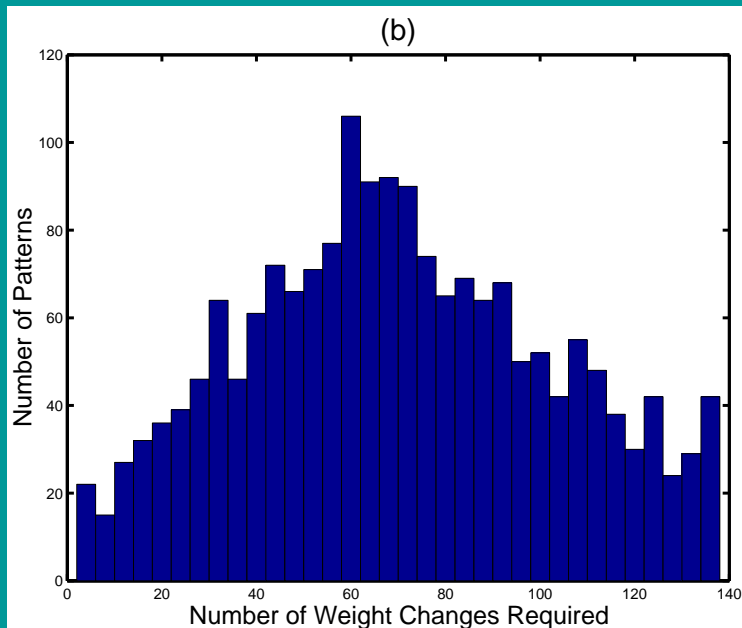
- a (adaptation) and l (size of initial weight distribution) are thought of as parameters

Convergence Rates III (results)

Rule	Theory		Simulation	
	Mean	St. Dev.	Mean	St. Dev.
(0,0,0,0)	26.74	24.61	26.11	23.73
(0,0,0,1)	75.72	39.67	75.89	39.79
(0,0,1,0)	39.33	12.05	44.23	34.67
(0,0,1,1)	39.66	31.97	40.08	31.96
(0,1,1,1)	36.68	28.89	37.54	29.29

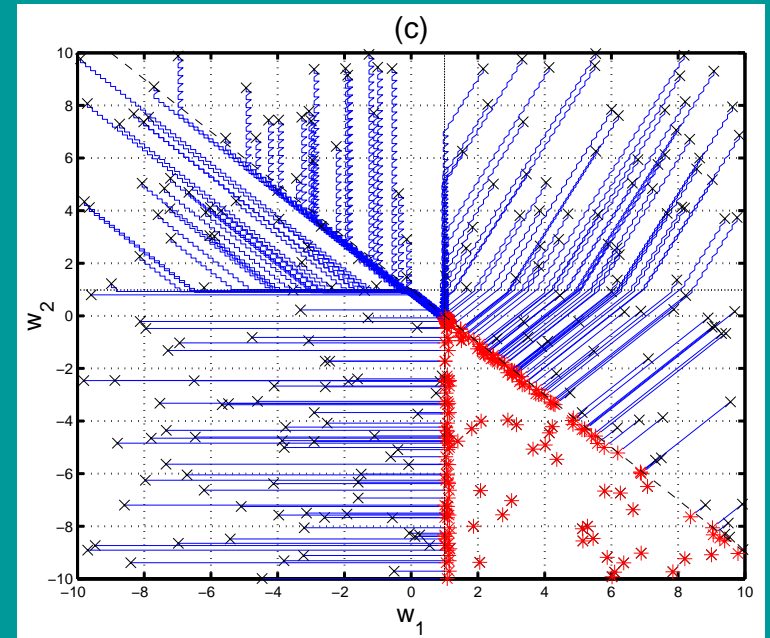
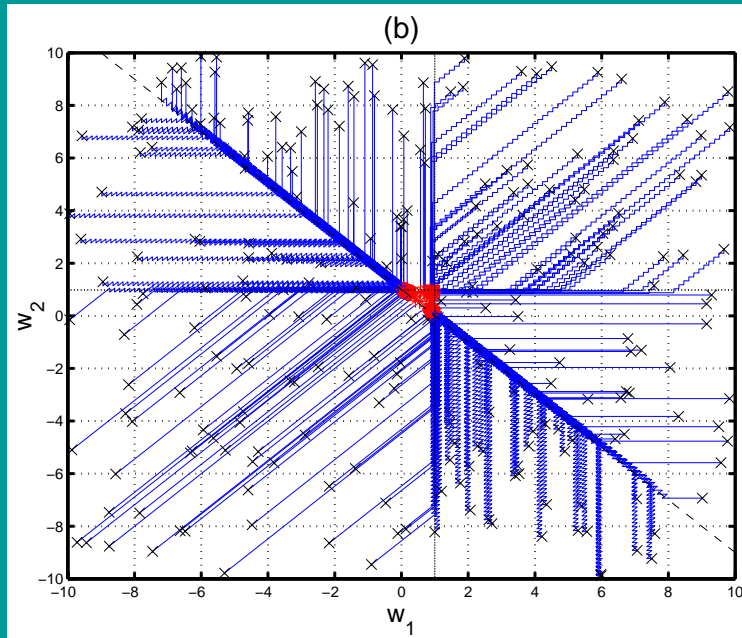
To compare note that e.g. the standard error of the mean for (0,0,0,0) would be $23.73/\sqrt{2000} = .53$.

Simulations I (weight histograms)



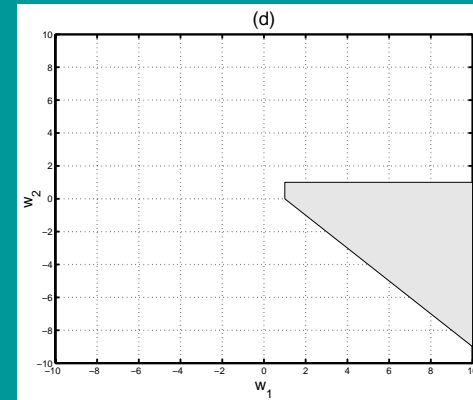
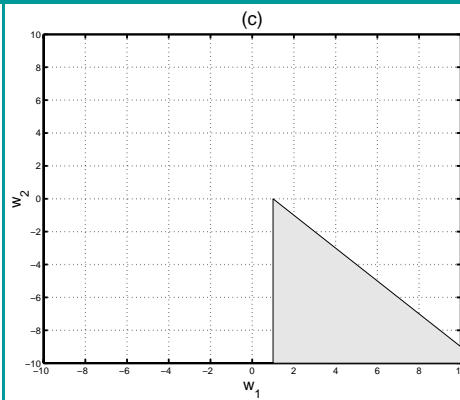
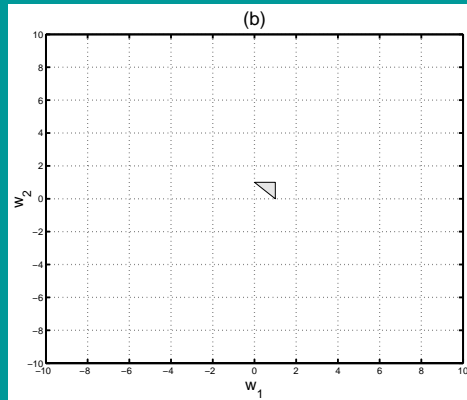
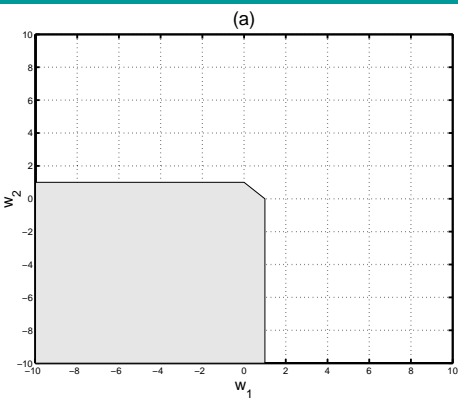
- Number of steps to converge for 2000 trials
- (b): $d^1=(0,0,0,1)$, (c): $d^3=(0,0,1,1)$
- Results for zero steps omitted for clarity

Simulations II (weight paths)



- Weight trajectories for 250 trials
- (b): $d^1=(0,0,0,1)$, (c): $d^3=(0,0,1,1)$

Simulations III (systems of inequalities)



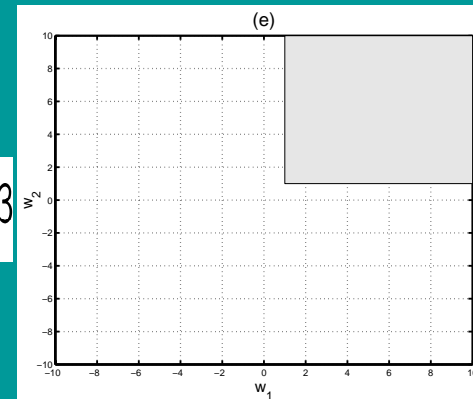
- The attractors in weight space for the five class representatives
- Solutions of the inequalities:

$$w_2(-1)^{d_1} \leq t(-1)^{d_1}, w_1(-1)^{d_2} \leq t(-1)^{d_2}, w_2(-1)^{d_3} \leq (-w_1 + t)(-1)^{d_3}$$

$$\mathbb{R}^2 \oplus \mathbb{R}^2$$

- For a perceptron (which the perceptron algorithm “solves”) thus partitions

$$\mathbb{R}^N \oplus \mathbb{R}^N$$



Computational Complexity

- P-hard problems can be solved in a number of steps equal to a polynomial in the size of the problem
- NP-hard problems can be verified as solvable in a number of steps equal to a polynomial in the size of the problem
- NP-complete problems are the hardest NP-hard problems
- The single-layer perceptron is trainable in polynomial time, but training the two-layer perceptron is NP-complete (Blum and Rivest, 1992)
- The equations given for the average convergence rates are polynomial in the size of the threshold t :

$$\mu_k(t; a, l) = \frac{1}{a} \sum_{j=0}^{N+1} a_j l^{N-1-j} t^j$$

- This suggests a possible different formulation of computational complexity for perceptron problems in terms of the weight dynamics and the size of the threshold

Purkinje Cell as Perceptron

- The main computational unit of the cerebellum, the Purkinje Cell, can be modelled as a perceptron (Brunel et. al., Neuron, 2004)
- Each PC receives several 100,000 inputs from granule cells
- Output goes to deep cerebellar nuclei
- PC's can be trained by classical conditioning
 - CS input via granule cells synapse on distal dendritic tree
 - US via climbing fibers synapse on soma and proximal processes to supervise learning

$$\mu_k\left(\frac{l}{a}, \frac{t}{l}\right) = \frac{l}{a} \sum_{j=0}^{N+1} a_j \left(\frac{t}{l}\right)^j$$

- A least squares fit could be used to find the coefficients a_j , where l = PC threshold, l =range of PC-GC synapse weights
- In this way, long term depression (LTD) and learning rate of cerebellum could be studied by varying PC physiological parameters

Conclusion

- To paraphrase Santayana: Even the behavior of the most complicated and elaborately realistic model for a complex system is bound to be reflected in the simplest possible model
- The simplest possible model permits us to examine how changing only a few parameters affect the behavior of a complex system
- The perceptron is such a simple model, and hence needs to be understood thoroughly as one of the conceptual building blocks of complex neural networks