Using the Perceptron Algorithm to Find Consistent Hypotheses

Martin Anthony
Department of Statistical and Mathematical Sciences
London School of Economics,
Houghton Street, London WC2A 2AE, UK.
m.anthony@lse.ac.uk.

John Shawe-Taylor
Department of Computer Science
Royal Holloway and Bedford New College
Egham Hill, Egham, Surrey TW20 0EX, UK.
john@dcs.rhbnc.ac.uk.

Abstract

The perceptron learning algorithm yields quite naturally an algorithm for finding a linearly separable boolean function consistent with a sample of such a function. Using the idea of a specifying sample, we give a simple proof that this algorithm is not efficient, in general.

A boolean function t defined on $\{0,1\}^n$ is *linearly separable* if there are $\alpha \in \mathbf{R}^n$ and $\theta \in \mathbf{R}$ such that

$$t(x) = \begin{cases} 1 & \text{if } \langle \alpha, x \rangle \geq \theta \\ 0 & \text{if } \langle \alpha, x \rangle < \theta, \end{cases}$$

where $\langle \alpha, x \rangle$ is the standard inner product of α and x. Given such α and θ, we say that t is represented by $[\alpha, \theta]$ and we write $t \leftarrow [\alpha, \theta]$. The vector α is known as the *weight-vector*, and θ is known as the *threshold*. This class of functions is the set of functions computable by the *simple boolean perceptron* (see [8, 9, 6]), and we shall denote it by BP_n.

We now give a fleeting description of the perceptron learning algorithm, and refer to [6, 1] for more details. For any *learning constant* $\nu > 0$, we have the *perceptron learning algorithm* L_ν, devised by Rosenblatt [8, 9], which acts sequentially as follows. Let t be any function in BP_n, which may be thought of as the *target*. The algorithm L_ν maintains at each stage a *current hypothesis*, which is updated on the basis of an example in $\{0,1\}^n$, presented together with its classification $t(x)$. (The initial hypothesis is some fixed 'simple' hypothesis. We shall take the initial hypothesis to have the all-0 vector as weight-vector, and threshold 0.) Suppose the current hypothesis is $h \leftarrow [\alpha, \theta]$ and that an example x is presented. Then the new hypothesis is $h' \leftarrow [\alpha', \theta']$ where

$$\alpha' = \alpha + \nu\,(t(x) - h(x))\,x, \quad \theta' = \theta - \nu\,(t(x) - h(x))\,.$$

The *Perceptron Convergence Theorem* [8, 6] asserts that no matter how many examples are presented, the algorithm makes only a finite number of changes, or updates (provided ν, which can be a function of n, is small enough).

As indicated in [3], given $t \in BP_n$ and a sample $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ of examples, we may use L_ν to find a linearly separable boolean function which agrees with t on $\mathbf{x}$—that is, which is *consistent* with t on $\mathbf{x}$. We simply keep cycling through x_1 to x_m in turn, until no updates are made in a complete cycle. Thus, the perceptron algorithm (for any learning constant ν) can be used as a *consistent-hypothesis-finder* (using terminology from [3]). A natural question is whether this is an efficient means of finding a consistent function. In fact, it is not, in the sense that the number of complete cycles required can be exponential in m, the size of the sample. This result appears to be accepted, but we have been unable to find a proof of it in the literature. We note that this is a very different result from those presented by Minsky and Papert[6] and Hampson and Volper [4] in their studies of the perceptron learning algorithm. Their results show that when the perceptron learning algorithm is used as an exact learning algorithm, the running time can be exponential in n, the domain dimension. Our result shows that, for fixed n, the running time of the related consistent-hypothesis-finder can be exponential in m, the number of examples presented. We remark that there is a polynomial time consistent-hypothesis-finder for BP_n: rephrase the problem as a linear programme and use Karmarkar's algorithm (see [3]). Thus the problem of finding a consistent hypothesis has no intrinsic difficulty.

We shall consider the boolean function f_{2n} of $2n$ variables with formula

$$f_{2n} = u_{2n} \wedge (u_{2n-1} \vee (u_{2n-2} \wedge (u_{2n-3} \vee (\ldots (u_2 \wedge u_1)) \ldots),$$

in the standard notation for describing boolean functions in terms of the literals $u_1, u_2,,$ the OR connective $\vee$ and the AND connective $\wedge$. This function, discussed in [7, 4, 5], is in BP_n. (Indeed, all such 'nested' functions are; see [2].) The following easily obtained result is along the lines of results due to Muroga [7].

Proposition 1 *Let n be any positive integer and suppose $f_{2n} \leftarrow [\alpha, \theta]$. Then $\alpha_{2n} \geq \sqrt{3}^{n-1} \min(\alpha_1, \alpha_2)$.* □

We have the following result, a special case of a more general 'specification' result from [2].

Proposition 2 *Let the set $S_n \subseteq \{0,1\}^{2n}$ of cardinality $2n+1$ be defined for each positive integer n as follows. $S_1 = \{(0,1),(1,0),(1,1)\}$, and, for $n \geq 1$,*

$$S_{n+1} = \{x01 : x \in S_n\} \cup \{(11\ldots10),(00\ldots011)\}.$$

Then the only function $h \in BP_n$ consistent with f_{2n} on S_n is f_{2n} itself. □

Combining these two results, we obtain the result we seek.

Theorem 3 *For any fixed $\nu > 0$, the consistent-hypothesis-finder arising from the perceptron learning algorithm L_ν does not always run in time polynomial in the size of its input.*

Proof: Suppose we take the target t to be f_{2n} and we take S_n as the input to the consistent-hypothesis-finder. Suppose the initial hypothesis is $h \leftarrow [(00,\ldots,0),0]$. Let N be the number of updates made before a consistent hypothesis is produced. By Proposition 2, this consistent hypothesis must be f_{2n} itself, and so if it is represented by $[\alpha,\theta]$, then $\alpha_1, \alpha_2 > 0$ and, by Proposition 1, $\alpha_{2n} \geq \sqrt{3}^{n-1}\min(\alpha_1,\alpha_2)$. After N updates, the maximum entry in the new weight-vector α' is at most $N\nu$ and the minimum entry is certainly at least ν. Hence the ratio of maximum entry to minimum entry is at most N. But, since the final output weight-vector has this ratio at least equal to $\alpha_{2n}/\min(\alpha_1,\alpha_2) > \sqrt{3}^{n-1}$, it follows that $N \geq \sqrt{3}^{n-1}$, which is exponential in n, and hence in $2n+1$, the size of the input. □

This result also holds if $\nu = \nu(n)$ is a function of n, bounded above by some constant. (Usually, this is certainly the case since ν is taken to be decreasing with n.)

References

[1] M. Anthony and N. Biggs, *Computational Learning Theory: An Introduction*, Cambridge University Press: Cambridge, UK, 1992.

[2] M. Anthony, G. Brightwell, D. Cohen and J. Shawe-Taylor, On exact specification by examples, in *COLT'92, Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, July 1992.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the ACM*, 36(4), 1989: 929–965.

[4] S.E. Hampson and D.J. Volper, Linear function neurons: structure and training. *Biological Cybernetics* 53, 1986: 203–217.

[5] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear threshold learning algorithm. *Machine Learning*, 2(4), 1988: 285–318.

[6] M. Minsky and S. Papert, *Perceptrons*. MIT Press, Cambridge, MA., 1969. (Expanded edition 1988.)

[7] S. Muroga, Lower bounds of the number of threshold functions and a maximum weight. *IEEE Transactions on Electronic Computers*, 14, 1965: 136–148.

[8] F. Rosenblatt, Two theorems of statistical separability in the perceptron. In *Mechanisation of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory, November 1958. Vol. 1*. HM Stationery Office, London, 1959.

[9] F. Rosenblatt, *Principles of Neurodynamics*. Spartan, New York, 1962.