6.858/18.428 Machine Learning **Lecture 3 : September 14, 1994**

Lecturer: Ron Rivest *Scribe: Ping Huang*

Outline

- Review of Winnow Algorithm
- Perceptrons
- VC-dimension and Mistake Bounds

3.1 Review of Winnow Algorithm

We established in the last lecture that the Winnow algorithm [2] can learn the concept class of monotone disjunctions $C_k = \{x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}\}$ in an on-line prediction model, making $O(k \lg n)$ mistakes when there are a total of n input variables. Note that the existence of such a mistakes bound does *not* guarantee that the learner ever successfully converges to exactly the right concept.

The Winnow algorithm can learn concepts which are linearly separable. That is, there exists a vector of weights w and threshold Θ such that $w \cdot x > \Theta$ if and only if x satisfies the target concept.

It can be shown that $O(k \lg n)$ is the best possible mistake bound on this class of problems, hence the Winnow algorithm is by this measure within a constant factor of the best possible algorithm.

The Winnow algorithm as we outlined it can only deal with a very restricted set of problems: monotone disjunctions of k variables. How can we extend it? One relatively simple extension might be to learn *arbitrary* disjunctions of k variables. We could do this by having $2n$ input variables for Winnow such that for $i = 1 \ldots n$, $x_{i'} = x_i$ and $x_{2i'} = \bar{x}_i$. Another more complex extension might be to learn the concept class k-DNF (disjunctive normal form with k literals or fewer per term); we can accomplish this by taking as input variables for the Winnow algorithm the $O(n^k)$ members of the set of all such possible conjunctions. Various other transformations exist to apply Winnow to other classes of functions.

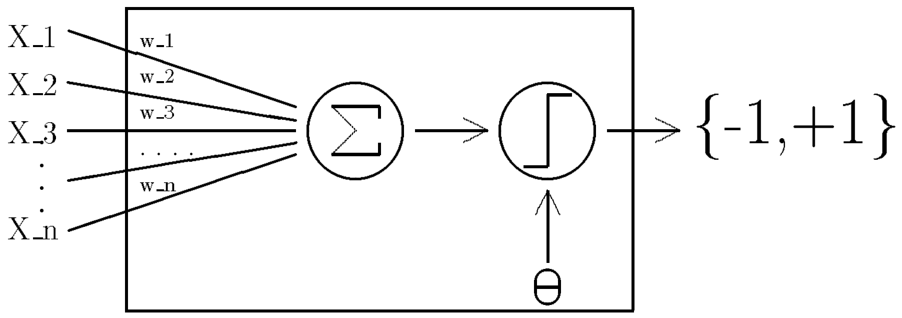

Figure 3.1: One way to visualize a perceptron as a device. If $w \cdot x > \Theta$, output 1, else output -1.

The fact that that the number of errors grows only logarithmically in n is very important. The Winnow algorithm does well on problems where n is large but k, the number of relevant attributes, is relatively small. How well does learning occur when n is infinite (as is the case for many real-world problems when instances are described very carefully in detail), or when the set of attributes is unknown beforehand? An infinite attributes model is discussed in [1].

3.2 Perceptrons

3.2.1 Introduction

A perceptron (see Figure 3.1) can be thought of as an engineering simplification of our understanding of how a real neuron works. It takes n inputs, multiplies by some weight vector w, compares the result with some threshold Θ, outputting either +1 or -1. This is more general than the concept class of the Winnow algorithm. Notably, the input variables are real numbers, not binary quantities; the weight vector components are also real numbers. Analysis therefore may be messier, since the concept class size is of infinite cardinality.

Perceptrons do fairly well in practice, even if their period of excitement has passed; they *can* learn many functions of interest, and are easy to train. They can model high-order (e.g., non-linear) effects easily — just add new variables. They are optimal Bayesian classifiers in many interesting cases.

How do you train a perceptron in the on-line prediction model? An algorithm is discussed in the next section.

3.2.2 Perceptron Convergence Algorithm and Theorem

We now set out to prove the perceptron convergence theorem. The perceptron training procedure is due to Frank Rosenblatt [4], and the proof of convergence we give is adapted from a proof given in Minsky and Papert's book *Perceptrons* [3]. First, we make some assumptions necessary to the proof.

Assumption 1 $\exists w, \Theta$ *such that* w, Θ *can* strictly *(i.e.,* $w \cdot x > \Theta$*, not just* $\geq$*) classify all examples correctly.*

Assumption 2 $\Theta = 0$ *(without loss of generality).* Justification: a non-zero Θ is just a bias. We could introduce an additional input $x_{n+1} = 1$ and set $w_{n+1} = -\Theta$, if necessary, to erase the bias. With this assumption, we can then focus entirely on adjusting the weight vector during the training process.

Assumption 3 *All examples are positive (without loss of generality).* Justification: we can transform all input negative examples by negating each input variable — if $w \cdot x < 0$ then $w \cdot (-x) > 0$, hence $-x$ will be a positive example if x was negative. (This uses the previous assumptions.)

Assumption 4 *No noise.* Thus, no mislabelled training examples. (The presence of noise is itself an entire sub-topic of this course.)

Perceptron Convergence Algorithm:

- Set w initially to $\langle 0, 0, \ldots, 0 \rangle$.
- Given an infinite sequence of transformed (hence all positive) examples $x^{(1)}, x^{(2)}, \ldots$: predict +1 if $w \cdot x > 0$, else predict -1.
- If an error occurs, set $w\prime = w + x^{(i)}$.

If a prediction was in error, it could only have been a false negative error. We therefore want to tweak w to make $w \cdot x$ more positive (accepting that perhaps it will take more than a single error to push the dot product over the threshold). Simple substitution

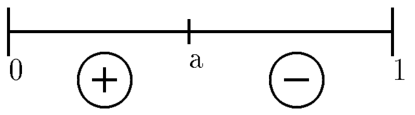

Figure 3.2: A simple concept class which nevertheless cannot be learned with a finite number of mistakes. The domain is $x \in \Re$, $0 < x < 1$; each concept c is characterized by some $a \in \Re$: a concept labels numbers less than or equal to its characteristic a positive and labels all other numbers negative.

and expansion gives $w\prime \cdot x = (w + x^{(i)}) \cdot x^{(i)} = w \cdot x + x^{(i)} \cdot x^{(i)}$. The lattermost term must be non-negative (being the magnitude of x squared), hence we meet the goal of making $w \cdot x$ more positive.

Note that we need to watch $|x|^2$. An adversarial choice of examples could cause non-convergence, e.g., $|x|^2$ values of subsequent examples could have geometrically decreasing magnitudes, such that any given example contributes more weight than the remainder of the series of examples. Therefore we need the next assumption.

Assumption 5 $|x| = 1$ *(without loss of generality).* This requires a transformation of the input variables. It is not justified here.

We need another assumption in order to prove anything about an upper bound on number of errors. Why it is necessary can be most easily demonstrated with a simple concept class (see Figure 3.2). Concepts of this class cannot be learned with a finite number of errors. A simple justification might be to observe that a potentially contains an infinite amount of information. More formally, it can be demonstrated that an adversary can easily force an infinite number of mistakes, by not choosing a in advance. At any stage in the learning process, the learner has some region of uncertainty where it believes a is, and the adversary can then present the learner with an x in this region, and whatever the learner guesses, tell the learner that it is wrong (thus narrowing the uncertainty region, but the region is infinite divisible).

Assumption 6 $(\exists\delta)(\exists v)$ *such that* $(\forall x^{(i)}) v \cdot x^{(i)} > \delta$. A concept c might be well-defined in the *delta* gap (see Figure 3.3, but this assumption gives the learner some leeway: it doesn't have to learn c perfectly, just well enough to correct discriminate the cases outside the gap. That is, hyperplanes AD or BC would also be good enough, as the perceptron does not need to converge to exactly hyper-plane BD.

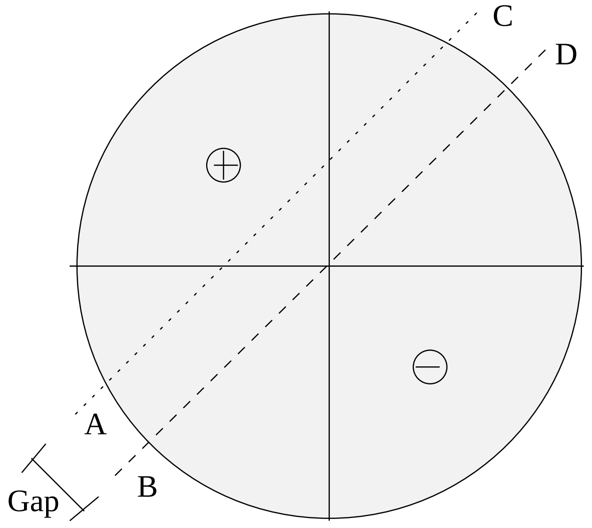

Figure 3.3: Visual aid in understanding the gap introduced by δ on a two-dimensional domain. BD represents the actual concept to be learned, but the learner only has to converge to a concept which lies within AC and BD within the unit circle.

Theorem 1 *(Perceptron Convergence Theorem) If all examples have unit length and are linearly separable (in such a way that there is a hyperplane which* strictly *separates positive and negative examples), then a learner which uses the Perceptron Convergence algorithm will only make a finite number of mistakes. That is, the training procedure converges.*

Proof:

Let $|v| = 1$ (without loss of generality), rescaling v if necessary. Let $f(w) = \frac{v \cdot w}{|w|} = \cos(\alpha) \leq 1$ (defining α as the angle between v and w). (Hopefully w converges on v, in which case $f(w)$ increases.) Except for the initial $w = (0, 0, \ldots, 0)$, $f(w)$ is well-defined. Initially, we have $f(w) \simeq 0$.

We update the weight: $w\prime = w + x$. How does $f(w)$ change when w is updated?

Numerator: $v \cdot w\prime = v \cdot (w + x) = v \cdot w + v \cdot x \geq v \cdot w + \delta$, since $v \cdot x > \delta$. Thus the numerator goes up by at least δ on each update.

Denominator: $|w\prime|^2 = |w|^2 + 2w \cdot x + |x|^2 \leq |w|^2 + 1$. Why? We know $2w \cdot x$ must be negative because an update of w only occurs with a false negative ($w \cdot x$ negative); and $|x|^2 = 1$ by assumption. Thus the square of the denominator goes up by at most 1.

After t updates (i.e., t mistakes), $f(w) > \frac{t\delta}{\sqrt{t}}$. We know that $f(w) \leq 1$, since it is the cosine of the angle between v and w. Thus $\frac{t\delta}{\sqrt{t}} \leq 1$. Simplifying gives $\sqrt{t}\delta \leq 1$, and thus $t \leq \frac{1}{\delta^2}$. So there can be at most $\frac{1}{\delta^2}$ mistakes.

■

Minsky's book on perceptrons helped to kill off the field because he lambasted perceptrons for being inefficient for hard classes. Today, that theoretical result might not have discouraged today's (more pragmatic) researchers as much as it did in 1969.

Even for classes over boolean variables, there exist cases where the δ gap must be exponentially small, i.e., $\delta \approx 2^{-cn}$. Then the perceptron convergence theorem results implies that there could be an exponential number of mistakes made.

The number of boolean threshold functions representable as $w \cdot x > \Theta$ is $2^{\Theta(n^2)}$. *If* we could implement the halving algorithm *efficiently* (but we usually can't...), we could learn with merely n^2 mistakes, theoretically considerably better than the perceptron's $O(\frac{1}{\delta^2})$ mistakes. The gap between practice and theory, however, is such that perceptron is a more practical technique, even if depends on this δ which is hard to characterize, and has a worse upper bound on the number of mistakes which might be made.

3.3 VC-dimension and Mistake Bounds

Until now, most of our analysis for learning algorithms has been to establish upper bounds. Are there method of establishing lower bounds? (This may perhaps be useful in putting a stop to theoreticians attempting to come up with new algorithms for some particular problem, if it can be shown that an existing algorithm is already within a constant factor of the best algorithm possible.) $\mathrm{VC}_{\mathtt{DIM}}$ is such a method; it attempts to measure the complexity of a concept class.

Definition 1 *$VC_{\mathtt{DIM}}(C)$ = the largest d such that there exist d distinct points $x_1, x_2, \ldots, x_d$ that can be labeled positive/negative in all 2^d ways. I.e., for any such labelling, there exists some concept $c \in C$ that gives those points such labels.*

Example 1 *For the number-line concept class in Figure 3.2, $VC_{\mathtt{DIM}}(C)=1$.* For $d = 1$, given point p, we can choose a concept from C to make the label for p either positive or negative. For $d = 2$, however, given points p_1, p_2, we cannot always choose a concept that labels p_1 positive and p_2 negative; an example of when no $c \in C$ exists for this labelling is when $p_1 > p_2$.

Example 2 *$VC_{\mathtt{DIM}}(C)=3$ for the concept class characterized by a line separating positive instances from negative instances on a two-dimensional plane.* For $d = 1$, $d = 2$, and $d = 3$, it is always possible to draw a line that gives d given points any permutation of labels. For $d = 4$, however, this is not possible; a simple counterexample is four points arranged in a square, where two of the points diagonal from each other are labelled positive, and the other two points are labelled negative.

Claim 1 *$VC_{\mathtt{DIM}}$(concept class of generalized half-spaces in $\Re^N$)$= N + 1$.* (Note that the number-line example above is *not* the completely general concept class of half spaces in $\Re^1$. In particular, if concepts where points *greater* than a were labelled positive were in C, then $\mathrm{VC}_{\mathtt{DIM}}$(C) would indeed be 2.)

Claim 2 *$VC_{\mathtt{DIM}}(C) \leq \lg |\mathtt{C}|$.* This comes from the 2^d in the definition for $\mathrm{VC}_{\mathtt{DIM}}$.

Claim 3 *$VC_{\mathtt{DIM}}(C) \leq opt(C)$.* Justification: an adversary is guaranteed by the definition of $\mathrm{VC}_{\mathtt{DIM}}$ that there is some c∈C such that it can present $x_1, x_2, \ldots, x_d$ examples to the learner, and every single time tell the learner that it was wrong.

References

[1] Avrim Blum, "Learning Boolean Functions in an Infinite Attribute Space". *Machine Learning*, October 1992, 9:4:373-386.

[2] Nick Littlestone: "Learning quickly when irrelevant attributes abound: A new linear-threshhold algorithm". *Machine Learning*, 2:285-318, 1988.

[3] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[4] F. Rosenblatt: "The Perceptron: A probabilistic model for information storage and organization in the brain". *Psychological Review*, 65:386-407, 1958. Article reprinted in *Neurocomputing* (MIT Press, 1988).