CS/ENGRI 172, Fall 2002
9/23/02: Lecture Eleven Handout

Topics: The perceptron learning algorithm; the perceptron convergence theorem.

Terminology reminder

Suppose we have a function f assigning values of 0 and 1. It is traditional to call an instance $\vec{x}$ *positive* if $f(\vec{x}) = 1$, and *negative* otherwise, since in the former case, $\vec{x}$ is an example of the concept f represents. (Think of 0 and 1 as "no", and "yes", if you like.)

To avoid confusion with negative and positive numbers, I'm trying to avoid using the common "positive/negative example" terminology, but since it's standard I tend to forget. Apologies for any confusion.

More properties of the inner product

It should be fairly easy to convince yourself that for vectors $\vec{v}$, $\vec{w}$, $\vec{y}$, and $\vec{z}$ of the same dimensionality, we have the following *distributive* property:

$$(\vec{v} + \vec{w}) \cdot (\vec{y} + \vec{z}) = \vec{v} \cdot \vec{y} + \vec{w} \cdot \vec{y} + \vec{v} \cdot \vec{z} + \vec{w} \cdot \vec{z}. \tag{1}$$

For example, in the two-dimensional case,

$$\begin{aligned}
(\vec{v} + \vec{w}) \cdot (\vec{y} + \vec{z}) &= (v_1 + w_1, v_2 + w_2) \cdot (y_1 + z_1, y_2 + z_2) \\
&= (v_1 + w_1)(y_1 + z_1) + (v_2 + w_2)(y_2 + z_2) \\
&= (v_1 y_1 + w_1 y_1 + v_1 z_1 + w_1 z_1) + (v_2 y_2 + w_2 y_2 + v_2 z_2 + w_2 z_2) \\
&= (v_1 y_1 + v_2 y_2) + (w_1 y_1 + w_2 y_2) + (v_1 z_1 + v_2 z_2) + (w_1 z_1 + w_2 z_2) \\
&= \vec{v} \cdot \vec{y} + \vec{w} \cdot \vec{y} + \vec{v} \cdot \vec{z} + \vec{w} \cdot \vec{z}
\end{aligned}$$

Equation 1 implies that

$$\vec{v} \cdot (\vec{y} + \vec{z}) = \vec{v} \cdot \vec{y} + \vec{v} \cdot \vec{z}.$$

Outline of our perceptron convergence theorem proof

We present a somewhat oblique, but therefore interesting, proof. The general ideas are as follows. Given all the constraints we have about the oracle and learner,

- Define a *score* function, indicating how close the learner's vector $\vec{w}$ is to a particular "target" vector $\vec{w}^*$. (Why the quotation marks?) Our particular score measure takes the form N/D (numerator over denominator), and starts at 0.
- Show that at each *update* of the perceptron learning algorithm, i.e., where $\vec{w}_{old}$ gets changed to $\vec{w}_{new}$, the score measure increases by a non-negligible amount:
  - The score numerator N increases by *at least* g, the *gap* quantity.
  - The square of the score denominator D increases by *at most* 1.

  Hence, after t updates, the score will have increased by at least $\sqrt{t}g$ from the initial score of 0.
- But, since our particular score measure is upper-bounded by one, we get that t can be at most $1/g^2$, which, since $g > 0$, implies only a finite number of updates gets made.