The road to wisdom? - Well, it’s plain and simple to express:
Err
and err
and err again
but less
and less
and less.

— a “grook” by Piet Hein

Due at the beginning of class on Wednesday, October 2. Hand in the parts separately, with your name on each part. Answers will be graded on both correctness and clarity. If a question asks for explanation, then no credit will be assigned for answers without explanation. In general, you should always include an explanation of your answers when appropriate (among other things, this helps in assigning partial credit).

Part A

In this part, we study simple ways in which several perceptrons can be combined to compute more complicated functions than single perceptrons alone are capable of computing. This is accomplished by structuring the perceptrons in a network, where the outputs of some of the perceptrons serve as the inputs to others. In particular, a useful function of binary vectors (i.e., those whose components are either zero or one) is the so-called “and” function; implementing “and” as a perceptron will allow us to combine individual perceptron outputs in meaningful ways.

1. Let AND be a perceptron with two-dimensional weight vector $\vec{w}_A = (.5, .5)$ and threshold $T_A = .7$. Show that AND computes an “and” function in that for the four binary inputs $(0, 0), (0, 1), (1, 0)$ and $(1, 1)$, it outputs a one for the last input vector and a zero for the other three (show all your work). You will thus demonstrate that for binary two-dimensional input vectors, AND fires iff\(^1\) the first input coordinate and the second input coordinate of the input vector are both 1.

2. Now we consider the small network of perceptrons pictured below. The input to the network consists of a two-dimensional vector $(x_1, x_2)$. The network is built out of:

   • a “left” perceptron with weight vector $\vec{w}_L = (2, -2)$ and threshold $T_L = 4$, taking $(x_1, x_2)$ as input;

   • a “right” perceptron with weight vector $\vec{w}_R = (3, -3)$ and threshold $T_R = 8$, also taking $(x_1, x_2)$ as input; and

   • an AND perceptron taking as input the vector $(f_L, f_R)$, where $f_L$ is the output of the “left” perceptron on input $(x_1, x_2)$ and $f_R$ is the output of the “right” perceptron on input $(x_1, x_2)$.

\(^1\)The notation “iff” stands for “if and only if”.
The output of the network is the output of the AND perceptron after \((x_1, x_2)\) has been fed into the network as input.

First, verify to yourself that this network outputs 1 if the input is \((16, 2)\) and outputs a 0 if the input is \((3, 4)\). (Don’t include these computations in your homework; rather, if you have trouble with the verification, please ask one of us about it.)

(a) As it turns out, although this network consists of three perceptrons, there is a single perceptron that computes precisely the same function.\(^2\) Give the weight vector and threshold of such a perceptron and explain why your perceptron computes the same function as the network. (To do so, you will need to specify what that function is.)

(b) Now consider the function

\[
 f(x_1, x_2) = \begin{cases} 
 1, & x_1 \geq 1 \text{ and } x_2 \geq 2; \\
 0, & \text{otherwise}.
\end{cases}
\]

This function is certainly not computable by a single perceptron, since, as shown here, it doesn’t correspond to a half-plane concept (we’ve truncated the 2-d plane for obvious space reasons):

Draw a perceptron network that computes \(f\), and explain why your network always computes the right values. Remember to include all the weight and threshold labels.

\(^2\)In fact, there are an infinite number of single perceptrons that are equivalent to this network.
Part B

3. In this problem, we study the perceptron learning algorithm’s updating behavior more carefully. Assume the usual restrictions on the oracle. As in class, let $\vec{w}^{(i)}$ denote the $i^{th}$ example produced by the oracle, where $i \geq 1$, and let $\vec{w}^{(i)}$ be the learner’s weight vector after seeing the $i^{th}$ example. Remember that it is possible for $\vec{w}^{(i-1)}$ and $\vec{w}^{(i)}$ to be the same.

In your solutions, please do not convert fractions (e.g. $1/2$) to decimals (e.g. 0.5) — this helps prevent minor mathematical errors from having nonobvious widespread effects on your answers (or, to put it another way, helps us tell the difference between a small math mistake and a misunderstanding of the overall concepts).

(a) Suppose that the learner has the weight vector $\vec{w}^{(i-1)}$, receives the example $\vec{x}^{(i)}$, and this example causes the learner to update its weight vector to a different vector $\vec{w}^{(i)}$ in the way specified by the perceptron learning algorithm. Show mathematically that $\vec{w}^{(i)} \cdot \vec{x}^{(i)} > \vec{w}^{(i-1)} \cdot \vec{x}^{(i)}$. (This means that in a sense $\vec{w}^{(i)}$ is “better” than $\vec{w}^{(i-1)}$ with respect to $\vec{x}^{(i)}$.) Be sure to explain each of your steps.

(b) Now, under the same conditions and notation of the previous subproblem, we ask, “are we guaranteed that after the update on account of $\vec{x}^{(i)}$, the learner’s perceptron assigns the correct label to $\vec{x}^{(i)}$?”. We demonstrate in this subproblem that the answer to this question is “no” by examining the following specific case.

Let $\vec{x}^{(100)} = (4/5, 3/5)$ and let $\vec{w}^{(99)} = (-2, 0)$. Show that the zero-threshold perceptron with weight vector $\vec{w}^{(99)}$ mislabels $\vec{x}^{(100)}$, and that after the update of the weight vector to $\vec{w}^{(100)}$ the learner’s perceptron still mislabels $\vec{x}^{(100)}$; be sure to show and explain your work.

(c) For the sake of argument, assume that in the previous subproblem you calculated that $\vec{w}^{(100)} = (-6/5, 3/5)$. Compute what $\vec{w}^{(102)}$ is according to the perceptron learning algorithm if the oracle now gives to the learner the input $(4/5, 3/5)$ twice in a row, that is, $\vec{x}^{(101)} = \vec{x}^{(102)} = (4/5, 3/5)$. Show and explain your work.

Part C

4. Here, we consider whether the perceptron learning algorithm can deal with oracles that are allowed to present example vectors not just on the unit $n$-dimensional sphere; that is, we examine the necessity of the instance-length restriction more carefully.

Suppose we modify the restrictions on the oracle and learner that we specified in class in the following way. Let $g > 0$ be the gap specified in the gap condition. Let $p$ and $q$ be two numbers such that $0 < p \leq g \leq q$. (We consider these numbers to be fixed, but not chosen by the learner.) We replace the old requirement that each example vector $\vec{x}^{(i)}$ presented by the oracle be of unit length by the following:

Every example $\vec{x}^{(i)}$ presented by the oracle satisfies

$$p \leq \text{length} \left( \vec{x}^{(i)} \right) \leq q.$$

All the other restrictions remain the same.

If we apply the perceptron learning algorithm from class in this new setting, is it still guaranteed to make a finite number of mistakes? If so, prove it, deriving an explicit bound on the number of mistakes in the manner we did in class. If not, explain clearly how an oracle can force the perceptron learning algorithm to make an infinite number of mistakes.