TERM PROJECT I MACHINE LEARNING

(second batch)

1. Create a simple domain consisting of a few boolean attributes that are labeled according to some simple function that guarantees separability. Create several versions of this domain, differing in the number of irrelevant attributes

   Implement perceptron learning and WINNOW. Compare them along the following two criteria: (1) the number of example-presentations needed to achieve 100% accuracy on the training set; (2) error rate measured on an independent testing set.

2. Create a domain of 100 training examples described by three numeric attributes, $x_i \in [0, 1]$. Label as **pos** those for which $1 - x_1 + x2 + x3$ is positive. Use perceptron learning with the following modification of the weight-updating rule:

   (a) $w_i = w_i + \eta[c(\mathbf{x}) - h(\mathbf{x})]x_i$

   (b) $w_i = w_i + \eta[c(\mathbf{x}) - \sum w_i x_i]x_i$

   (c) $w_i = w_i + \eta[c(\mathbf{x}) - \sum w_i x_i]$

   Will all of them converge to zero error rate on the training set? Compare the speed of conversion.

3. Create a set where each example is described by boolean attributes, $x_i, \ldots, x_6$ (plus varying numbers of irrelevant attributes), and labeled with the following classes:

   (a) C_1: at least five attributes have $x_i = 1$.

   (b) C_2: three or four attributes have $x_i = 1$.

   (c) C_3: two attributes have $x_i = 1$.

   (d) C_4: one attribute has $x_i = 1$.

   Apply perceptron learning to each of the four classes. See if the zero error rate on the training set can be achieved. Measure also error rate on an independent testing set.

4. Run induction of linear classifiers on selected boolean domains from the UCI repository[1] and compare the results in terms of error rates and computational costs (the latter expressed in terms of the number of example-presentations).

5. Experimenting with selected domains from the UCI repository, compare the behavior of linear and polynomial classifiers (using the perceptron learning). Observe how the former wins in simple domains, and the latter in highly non-linear domains.

6. Create a simple synthetic domain of your own. Then create a few versions of this domain, each with a different amount of class-label noice. Implement perceptrong learning and WINNOW, and apply these two algorithms to these data. Compare them in terms of how each of them "suffers" when there is noise in the data.

[1] `www.ics.uci.edu/~mlearn/MLRepository.html`