Problem Set 6: Learning via Perceptron

Due Oct 21 2024

Quan Wen

1 The Peceptron Algorithm

In class, we discussed the Perceptron learning algorithm, which is guaranteed to find a decision plane to separate two linearly-separable groups of points. In this exercise, you will write a Matlab code to implement the algorithm and use it to analyze numerically the perceptron capacity by using the threshold activation function studied in class.

The perceptron update rule changes the decision plane by cycling through the P given data-points one at a time and rotating the plane slightly when misclassifying a sample. The rule can be for-mulated as follows at time t.

$$If\ (w^{(t)} \cdot x^{\mu}) y_0^{\mu} < 0 \rightarrow w^{(t+1)} = w^{(t)} + \eta y_0^{\mu} x^{\mu}. \tag{1}$$

where $x^{\mu}, w \in \mathbb{R}^N$, $\mu = 1 \ldots P$, and $y_0^{\mu} = -1, 1$.

The weights vector w can start from any position and η (the learning rate parameter) can be any value and this algorithm will

converge to a solution in a finite number of steps (assuming the data is indeed linearly seperable).

1.1 Implement in MATLAB function,

$$function\,[w, converged, epochs] = perceptron(x, y0)$$

The function receives a matrix X (of size $N \times P$) of the samples and a vector y_0 (of size $P \times 1$) of the corresponding labels. The function should return the final weight vector w (of size $N \times 1$), a flag indicating if the algorithm converged and the number of epochs. In each epoch, we go over a cycle of P different input samples. Some of the inputs will be misclassified and the weight vectors will be updated according the learning rule specified by Eq. (1). The algorithm converges if no inputs are misclassifed in the last P input samples. The total number of epochs is just given by the total number of the input samples that have been checked during iterations divided by P. Set in your function a maximum number of epochs, which once exceeded the function terminates with a non-convergence indication. In the next sections you will use the algorithm to explore the convergence abilities and perceptron capacity. You'll do it by generating random binary inputs $x_i^\mu \in [-1, 1]$ and outputs. Run your algorithm and check convergence. Repeat each combination of N, P several times to collect statistics (e.g., $n = 50$).

1.2 Compare number of epochs till convergence for $P = 10$ and $N = 10, 20, 100$. How do the changes in N affect the convergence speed? Explain.

1.3 Define $\alpha = P/N$. Plot graphs of convergence probability as a function of $0 < \alpha < 3$. Discuss your results and compare them to results of Cover's function counting theorem that were derived in class.

1.4 One of the methods to grade different hyperplanes that correctly separate the training data, is to check their distance from the training points. We'll expect hyperplanes with a higher distance to have a smaller generalization error. The margin δ of a hyperplane is defined as the minimal distance from it to the set of training points:$\delta^* = min_\mu[(w \cdot x^\mu)y_0^\mu]$. Calculate the mean margin for different (P,N) combinations you used to evaluate convergence probability in **1.3** (naturally, you'll use the statistics from converged runs). Use the mean margin to calculate a margin histogram $H(P, N)$. Use the histogram to discuss - how do changes in P,N affect the hyperplane margin?