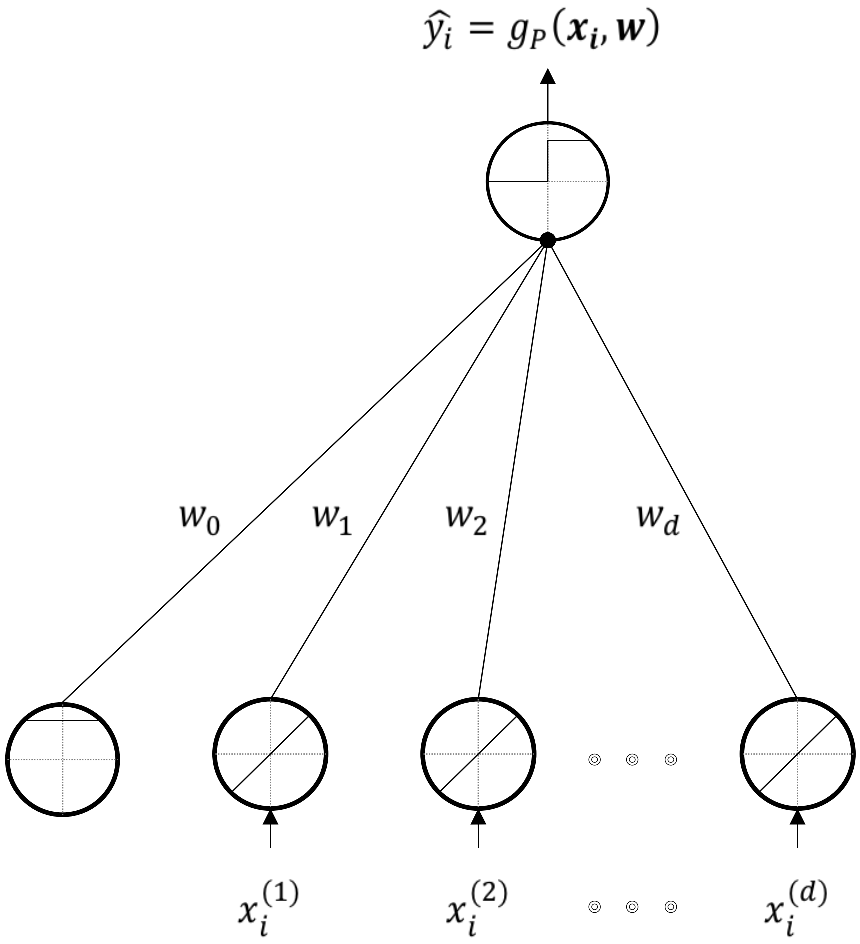
Learning in the Perceptron

- "Learning": how does the neuron adapt its weights in response to the inputs?



The Perceptron Learning Algorithm

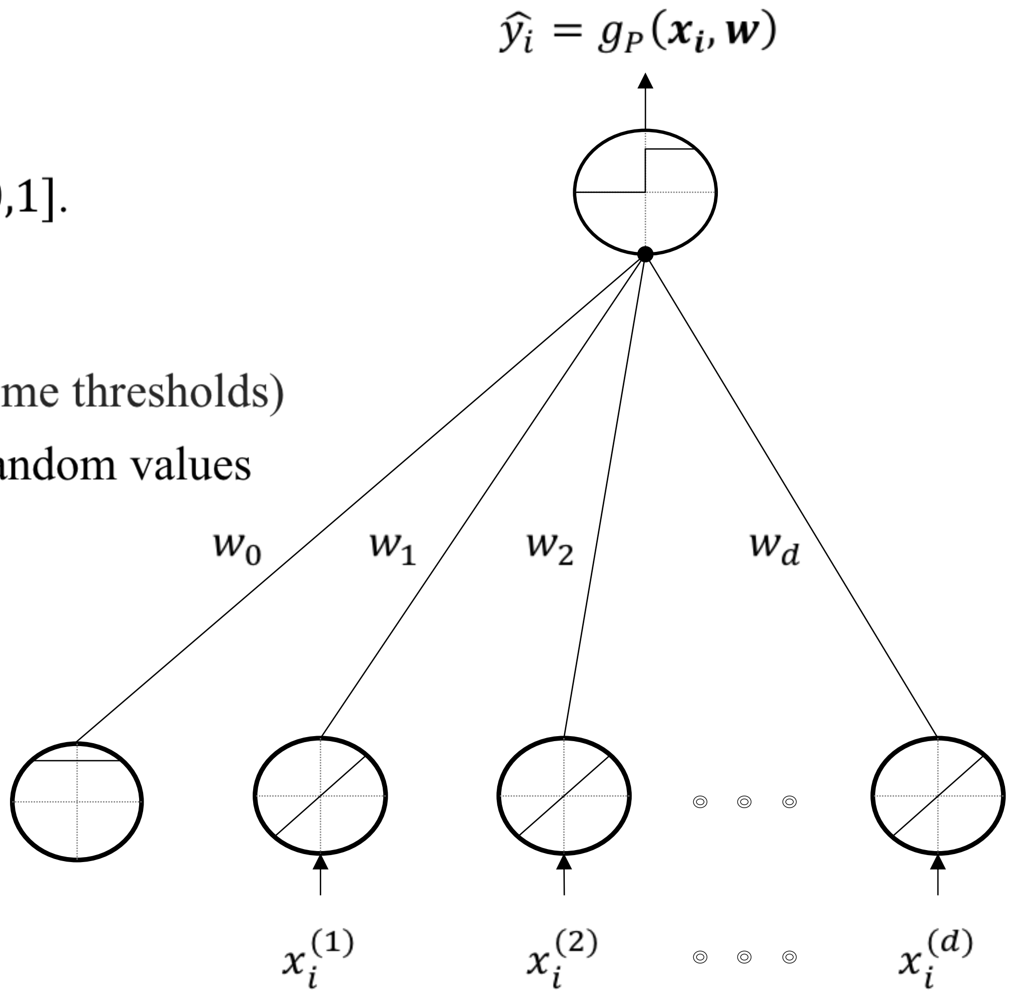
Input

- Training set

$D = \{(\boldsymbol{x}_i, y_i), i \in [1,2,\dots n]\}, y_i = [0,1].$

Initialization

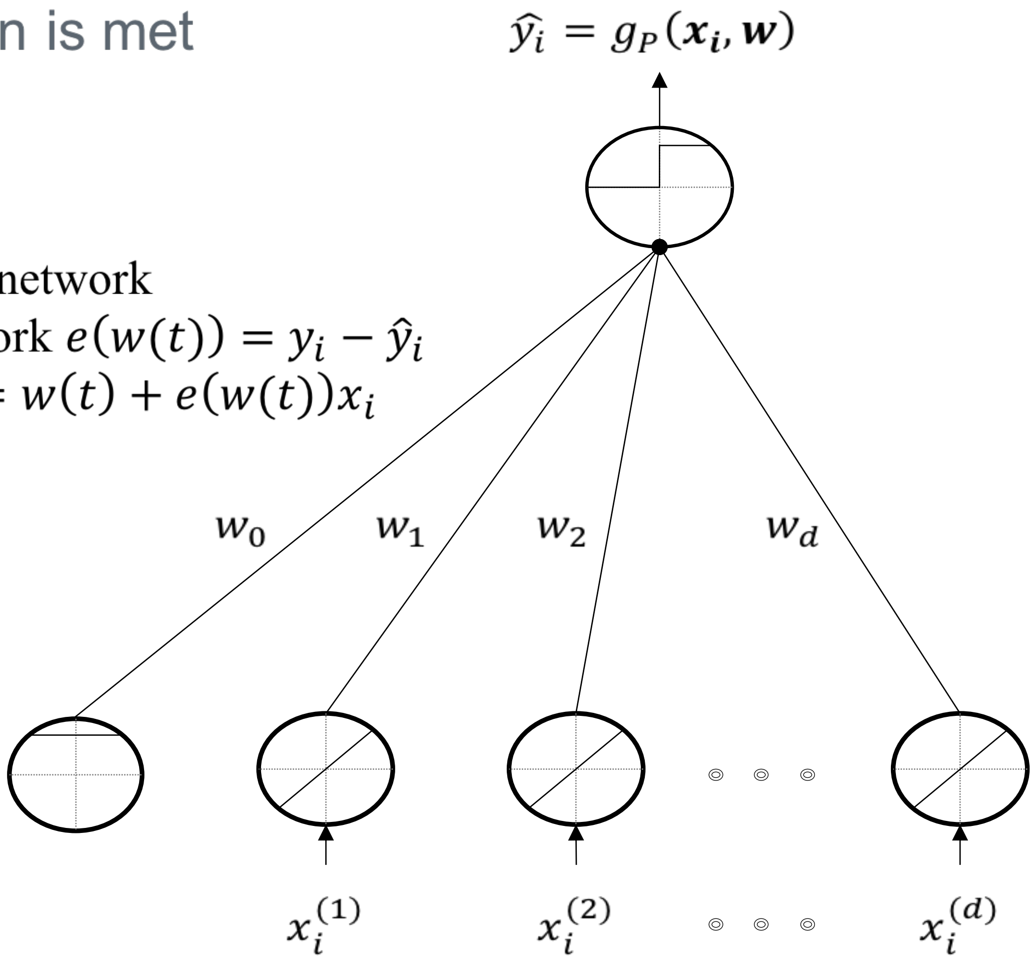
- Initialize the weights $\boldsymbol{w}(0)$ (and some thresholds)
- Weights may be set to 0 or small random values



The Perceptron Learning Algorithm (cont'd)

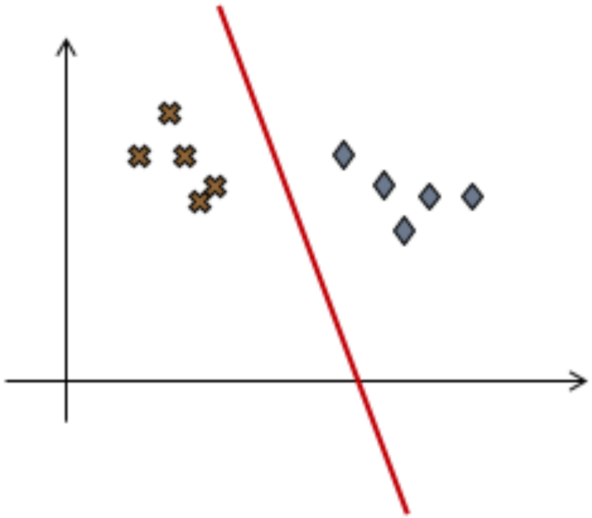
Iterate for *t* until a stop criterion is met

{
  for each sample x_i with label y_i:
  {
    compute the output $\hat{y}_i$ of the network
    estimate the error of the network $e(w(t)) = y_i - \hat{y}_i$
    update the weight $w(t+1) = w(t) + e(w(t))x_i$
  }
  t++
}



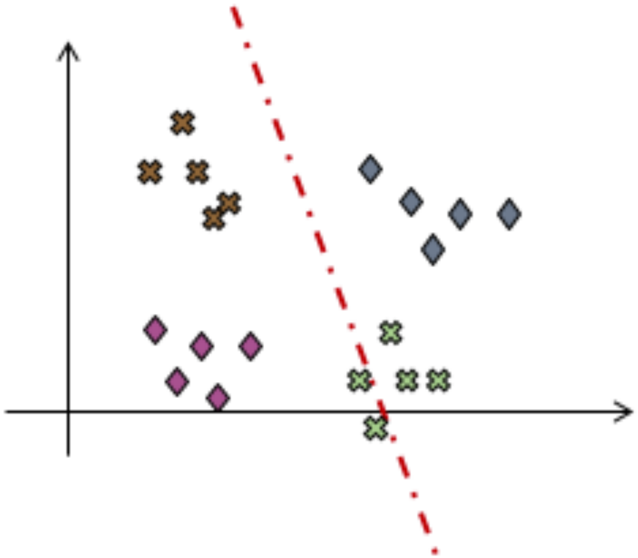
The Need for Multiple Layers

This is easy and can be learned by the Perceptron Algorithm

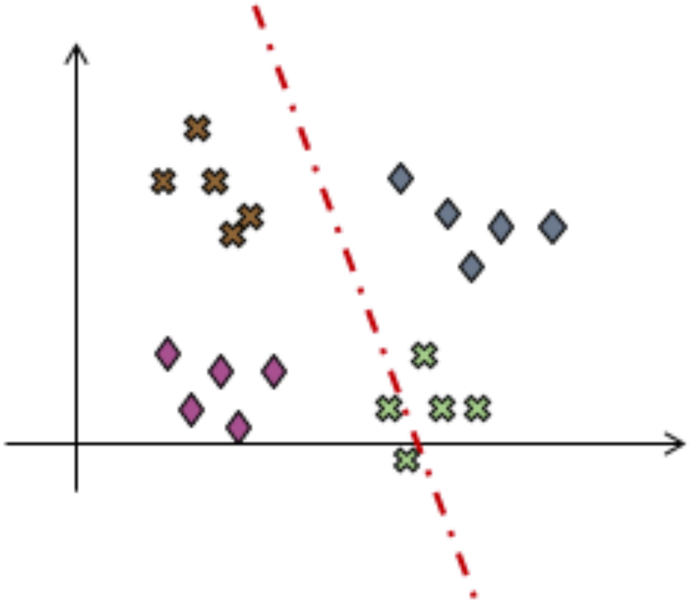


$$w_1x_1 + w_2x_2 + w_0 = 0$$

But how about this?



Extending to Multi-layer Neural Networks



The XOR Problem

Question: Can a multi-layer version of the Perceptron (MLP) help solving the XOR problem?

An MLP Solving the XOR Problem