1.2 Perceptron

Delta learning perceptron algorithm:

Step 1: Choose η , let E = 0, Initialize random weights (w).

Step 2: Calculate the output of the neural network.

$$y = step(w(k)^T x(k))$$

Step 3: Update weights using delta law.

$$w_i(k+1) = w_i(k) + \eta\left(d_i - y_i\right).x(k)$$

Step 4: Calculate error.

$$E = E + \frac{1}{2}\left\| d(k) - y(k) \right\|^2$$

Step 5: if *k* < *data_length* then *k* = *k+1*, go to *Step 2;* If *k* == *data_length*, go to Step 6.

Step 6: If E == 0, end of training; If E >0, let k=0, E=0 then go to Step 2.

Adaline

Delta learning Adaline algorithm:

Step 1: Choose η , let E = 0, Initialize random weights (w).

Step 2: Calculate the output of the neural network.

$$y = w(k)^T x(k)$$

Step 3: Update weights using delta law.

$$w_i(k+1) = w_i(k) + \eta\left(d_i - y_i\right).x(k)$$

Step 4: Calculate error.

$$E = E + \frac{1}{2}\left\| d(k) - y(k) \right\|^2$$

Step 5: if *k* < *data_length* then *k* = *k+1*, go to *Step 2;* If *k* == *data_length,* go to Step 6.

Step 6: If $E < \varepsilon$, end of training; If $E \geq \varepsilon$, let k=0, E=0 then go to Step 2.

1.3 Multi-Layer Perceptron

The learning algorithm : Back propagation

Step 1: Choose η , let E = 0, Initialize random weights (w, v).

Step 2: Calculate the output of the neural network.

Hidden layer: $net_h(k) = v^T x(k)$

$$y_h(k) = a_h(net_h(k))$$

Output layer: $net_o(k) = w^T y_h(k)$

$$y(k) = a_o(net_o(k))$$

Step 3: Calculate error.

$$E = E + \frac{1}{2}\|d(k) - y(k)\|^2$$

1.3 Multi-Layer Perceptron

The learning algorithm : Back propagation

Step 4: Update weights of output layer.

$$w_{(:,i)}(k+1) = w_{(:,i)}(k) + \eta \frac{-\partial E}{\partial w_{(:,i)}}, \; i : 0 \rightarrow output\_number - 1$$

Step 5. Update weights of hidden layer

$$v_{(:,j)}(k+1) = v_{(:,j)}(k) + \eta \frac{-\partial E}{\partial v_{(:,j)}}, \; j : 0 \rightarrow hidden\_number - 1$$

Step 5: if *k* < *data_length* then *k* = *k+1*, go to *Step 2;* If *k* == *data_length*, go to Step 6.

Step 6: If $E < \varepsilon$, end of training cycle; If $E \geq \varepsilon$, let k=0, E=0 then go to Step 2.

Learning Algorithm 2: the output of RBF

Step 1: - Choose η, E_{max} ;

- Initialize random weights $w_{iq}(i=\overline{1,n};\, q=\overline{1,l})$

; E=0.

Step 2: Calculate the output of the neural network

$$y_i = \sum_{q=1}^{l} \mathrm{w}_{iq}\varphi_q(x) = \sum_{q=1}^{l} \mathrm{w}_{iq} \exp(-\frac{\left\| x-\mu_q \right\|^2}{2\sigma_q^2})$$

Step 3: Update weights of the output layer

$$w_{iq}(k+1) = w_{iq}(k) + \eta\left(d_i(k) - y_i(k)\right).z_q(k); \quad i=\overline{1,n}; \quad q=\overline{1,l};$$

$$z_q(k) = \exp(-\frac{\left\| x(k)-\mu_q \right\|^2}{2\sigma_q^2})$$

The learning algorithm : Back propagation

Step 4: Calculate error:

$$E = E + \frac{1}{2}\sum_{i=1}^{n}\left(d_i(k) - y_i(k)\right)^2$$

Step 5: If k< data_length then k=k+1, go to step 2. If k= data_length then go to step 6.

Step 6: If E< E_{max} then end of training cycle. If E>= E_{max} then lets E=0, k=0 and go to Step 2.