Remarks on the **Linear Gradient descent** algorithm:

1. Stochastic version

Stochastic Gradient Descent Algorithm_Linear

Step 1 Input training data set:

D = $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ // N training examples

Step 2 Initialize **w** and choose a learning rate η.

// $\mathbf{w} = (w_0, w_1, ..., w_M)$; initialize all weights w_i to random values

Step 3 UNTIL a termination condition is met, DO

Step 4 FOR each training example $\mathbf{x}_k \in D$

// Update the weight vector for each training example

$\mathbf{w} \leftarrow \mathbf{w} + \eta(y_k - \widehat{y_k})\mathbf{x}_k$

// where $\widehat{y_k} = \mathbf{w}^T\mathbf{x}_k = [w_0, w_1, ..., w_M]\begin{bmatrix} x_1 \equiv 1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$

// $= w_0 x_0 + w_1 x_1 + ... + w_M x_M$

// : Linear Neuron

2. Batch version

Batch Gradient Descent Algorithm_Linear

Step 1 Input training data set:

D = $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ // N training examples

Step 2 Initialize **w** and choose a learning rate η.

// $\mathbf{w} = (w_0, w_1, ..., w_M)$; initialize all weights w_i to random values

Step 3 UNTIL a termination condition is met, DO

Step 4 Delta_ $\mathbf{w} = \mathbf{0}$ // Set up a zero column-vector with (M+1) components

Step 5 FOR each training example $\mathbf{x}_k \in D$

Delta_ $\mathbf{w} \leftarrow$ Delta_ $\mathbf{w} + \eta(y_k - \widehat{y_k})\mathbf{x}_k$

//where $\widehat{y_k} = \mathbf{w}^T\mathbf{x}_k = [w_0, w_1, ..., w_M]\begin{bmatrix} x_1 \equiv 1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$

// $= w_0 x_0 + w_1 x_1 + ... + w_M x_M$

// : Linear Neuron

Step 6 $\mathbf{w} \leftarrow \mathbf{w}$ + Delta_ **w** // Update the weight vector once per epoch

3. Stopping criteria of the **Gradient descent** algorithm usually includes:

(a) Stop when a maximum number of epochs has been exceeded.

(b) Stop when some error measure on the training set is small enough, say $\tau = 10^{-2}$.

For example:

Let y_k = dependent variable, and $\widehat{y_k}$ = estimated value = $\mathbf{w}^{\mathrm{T}}\mathbf{x}_{\mathrm{k}}$.

Then

(i) Mean Squared Error **(**MSE**)** = $\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{k=1}}^{\boldsymbol{N}}(y_k - \widehat{y_k})^2 < \tau$

(ii) Root Mean Squared Error **(**RMSE**)** = $\sqrt{\mathrm{MSE}} < \tau$

(iii) Mean Absolute Error **(**MAE**)** = $\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{k=1}}^{\boldsymbol{N}}|y_k - \widehat{y_k}| < \tau$

(Note: To avoid an infinite loop, you should have stop criteria (a) when you implement a program.)