Lecture 11.
MLP (III): Back-Propagation

Outline

- General cost function
- Momentum term
- Update output layer weights
- Update internal layers weights
- Error back-propagation

General Cost Function

$$E = \sum_{k=1}^{K} \sum_{i=1}^{N(L)} [e_i(k)]^2 = \sum_{k=1}^{K} \sum_{i=1}^{N(L)} [d_i(k) - z_i(k)]^2$$

$1 \le k \le K$ (K: # inputs/epoch); $1 \le K \le$ # training samples

i: sum over all output layer neurons.

N(ℓ): # of neurons in ℓ^{th} layer. $\ell = L$ for output layer.

Objective: Finding optimal weights that minimize *E*.

Approach: Use Steepest descent gradient learning, similar to the single neuron error correcting learning, but with multiple layers of neurons.

Gradient Based Learning

Gradient based weight updating with momentum —

$$\underline{w}(t+1) = \underline{w}(t) - \eta \nabla_{\underline{w}(t)} \mathbf{E} + \mu(\underline{w}(t) - \underline{w}(t-1))$$

η: learning rate (step size),

μ: momentum $(0 \le \mu < 1)$

t: epoch index.

Define: $v(t) = w(t) - w(t-1)$ then

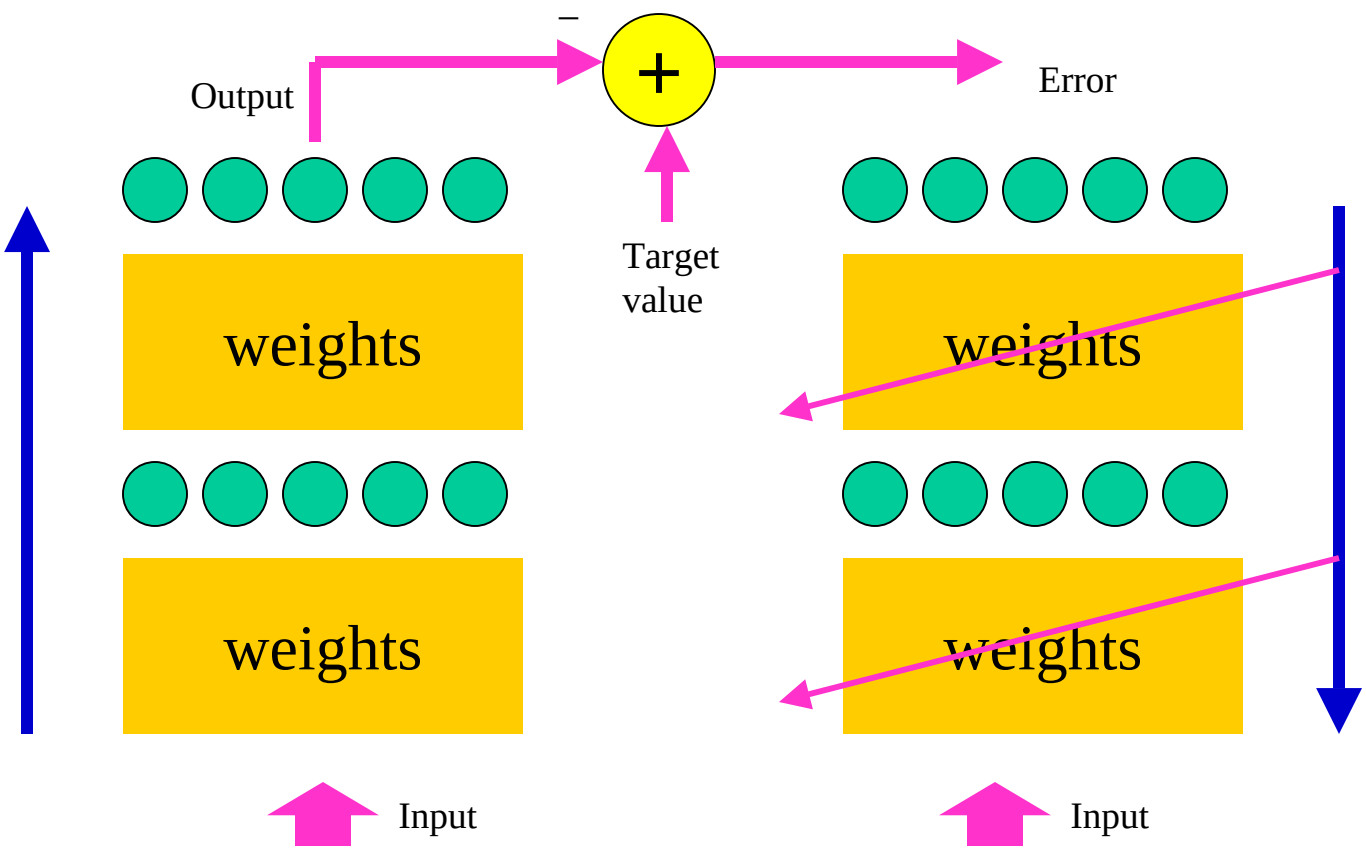
$$v(t+1) = \mu \cdot v(t) - \eta \cdot g(t)$$

$$= \mu^{t+1} \cdot v(0) - \eta \cdot \sum_{m=0}^{t} \mu^{t-m} g(m)$$

Momentum

- Momentum term computes an exponentially weighted average of past gradients.
- If all past gradients in the same direction, momentum results in increase of step size. If gradient directions changes violently, momentum reduces gradient changes.

Training Passes



Training Scenario

- Training is performed by “epochs”. During each epoch, the weights will be updated once.
- At the beginning of an epoch, one or more (or even the entire set of) training samples will be fed into the network. The feed-forward pass will compute output using present weight values and the least square error will be computed.
- Starting from the output layer, the error will be *back-propagated* toward the input layer. The error term is called the δ-error.
- Using the δ-error and the hidden node output, the weight values are updated using the gradient descent formula with momentum.

Updating Output Weights

Weight Updating Formula — error-correcting Learning

Weights are fixed over entire epoch. Hence we drop the index t on the weight: $w_{ij}(t) = w_{ij}$

For weights w_{ij} connecting to the output layer, we have

$$-\frac{\partial E}{\partial w_{ij}^{(L)}} = -\frac{\partial E}{\partial z_i^{(L)}(k)} \frac{\partial z_i^{(L)}(k)}{\partial w_{ij}^{(L)}}$$

$$= \sum_k [d_i(k) - z_i^{(L)}(k)] f'[u_i^{(L)}(k)] \frac{\partial u_i^{(L)}(k)}{\partial w_{ij}^{(L)}} = \sum_k \delta_i^{(L)}(k) z_j^{(L-1)}(k)$$

Where the δ-error is defined as

$$\delta_i^{(L)}(k) \equiv \frac{\partial E}{\partial u_i^{(L)}(k)} = [d_i(k) - z_i^{(L)}(k)] f'[u_i^{(L)}(k)]$$

Updating Internal Weights

- For weight $w_{ij}^{(\ell)}$ connecting ℓ–1th and ℓ th layer ($\ell \geq 1$), similar formula can be derived:

$$-\frac{\partial E}{\partial w_{ij}^{(\ell)}} = -\sum_{k=1}^{K} \frac{\partial E}{\partial u_i^{(\ell)}(k)} \frac{\partial u_i^{(\ell)}(k)}{\partial w_{ij}^{(\ell)}} = \sum_{k=1}^{K} \delta_i^{(\ell)}(k) z_j^{(\ell-1)}(k)$$

$1 \leq i \leq N(\ell)$, $0 \leq j \leq N(\ell-1)$ with $z_0^{(\ell-1)}(k) = 1$.

Here the <u>delta error</u> for internal layer is also defined as

$$\delta_i^{(\ell)}(k) = \frac{\partial E}{\partial u_i^{(\ell)}(k)}$$

Delta Error Back Propagation

For ℓ = L, as derived earlier,

$$\delta_i^{(L)}(k) = \frac{\partial E}{\partial u_i^{(L)}(k)} = f'[u_i^{(L)}(k)] \cdot [d_i(k) - z_i^{(L)}(k)]$$

For ℓ < L, $\delta_i^{(\ell)}(k)$ can be computed iteratively from the delta error of an upper layer, $\delta_m^{(\ell+1)}(k)$:

$$\delta_i^{(\ell)}(k) = \frac{\partial E}{\partial u_i^{(\ell)}(k)} = \frac{\partial E}{\partial z_i^{(\ell)}(k)} \cdot \frac{\partial z_i^{(\ell)}(k)}{\partial u_i^{(\ell)}(k)} = \frac{\partial E}{\partial z_i^{(\ell)}(k)} f'\left(u_i^{(\ell)}(k)\right)$$

$$= f'\left(u_i^{(\ell)}(k)\right) \cdot \sum_{m=1}^{N(\ell+1)} \frac{\partial E}{\partial u_m^{(\ell+1)}(k)} \cdot \frac{\partial u_m^{(\ell+1)}(k)}{\partial z_i^{(\ell)}(k)}$$

Error Back Propagation (Cont'd)

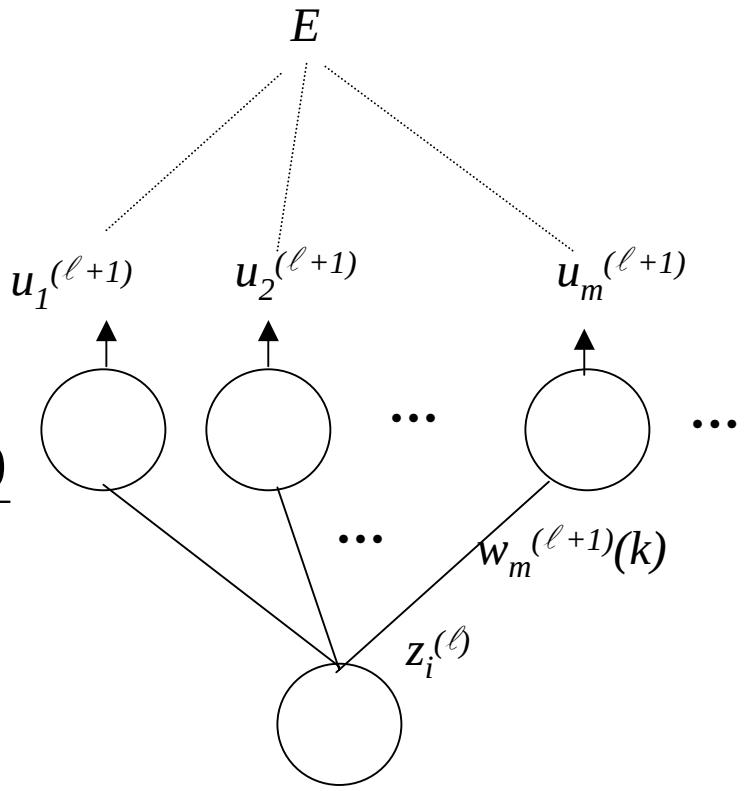
Note that for $1 \le m \le N$

$$u_m^{(\ell+1)}(k) = \sum_{i=0}^{N(\ell)} w_{mj}^{(\ell+1)} z_i^{(\ell)}(k)$$

Hence,

$$\delta_i^{(\ell)}(k) = f'\left(u_i^{(\ell)}(k)\right) \cdot \sum_{m=1}^{N(\ell+1)} \delta_m^{(\ell+1)}(k) \cdot \frac{\partial u_m^{(\ell+1)}(k)}{\partial z_i^{(\ell)}(k)}$$

$$= f'\left(u_i^{(\ell)}(k)\right) \cdot \sum_{m=1}^{N(\ell+1)} \delta_m^{(\ell+1)}(k) \cdot w_{im}^{(\ell+1)}$$



Summary of Equations (per epoch)

- Feed-forward pass: $z_0^{(\ell-1)}(k) \equiv 1$

  For *k = 1* to *K*, ℓ *= 1* to *L*, *i = 1* to *N*(ℓ),

$$z_i^{(\ell)}(k) = f(u_i^{(\ell)}(k)) = 1/(1+\exp[-u_i^{(\ell)}(k)])$$

$$u_i^{(\ell)}(k) = \sum_{j=0}^{N} w_{ij}^{(\ell)}(t) z_j^{(\ell-1)}(k)$$

t: epoch index

k: sample index

- Error-back-propagation pass:

  For *k = 1* to *K*, ℓ *= 1* to *L*, *i = 1* to *N*(ℓ),

$$\delta_i^{(\ell)}(k) = \begin{cases} f'\left(u_i^{(\ell)}(k)\right) \cdot \sum_{m=1}^{N(\ell+1)} \delta_m^{(\ell+1)}(k) \cdot w_{im}^{(\ell+1)}(t) & \ell < L, \\ f'\left(u_i^{(L)}(k)\right) \cdot [d_i(k) - z_i^{(L)}(k)] & \ell = L. \end{cases}$$

Summary of Equations (cont'd)

- Weight update pass:

  For $k = 1$ to K, $\ell = 1$ to L, $i = 1$ to $N(\ell)$,

$$-\frac{\partial E}{\partial w_{ij}^{(\ell)}(t)} = \sum_{k=1}^{K} \delta_i^{(\ell)}(k) z_j^{(\ell-1)}(k)$$

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - \mu \frac{\partial E}{\partial w_{ij}^{(\ell)}(t)} + \mu\left(w_{ij}^{(\ell)}(t) - w_{ij}^{(\ell)}(t-1)\right)$$