Lecture 15 : Stochastic Gradient Descent + Neural Networks

Stochastic Gradient Descent

Recall gradient descent:

$$\underline{w}^* = \arg\min_{\underline{w}} f(\underline{w})$$

$$\underline{w}^{(k+1)} = \underline{w}^{(k)} - \tau \nabla f(\underline{w}^{(k)})$$

Imagine $f(\underline{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\underline{w})$

e.g. if $f(\underline{w}) = \frac{1}{n}\|y - X\underline{w}\|_2^2$

$$= \frac{1}{n}\sum_{i=1}^{n} (y_i - \langle x_i, \underline{w}\rangle)^2$$

$$\Rightarrow f_i(\underline{w}) = (y_i - \langle \underline{x}_i, \underline{w}\rangle)^2$$

Then Gradient Descent =

$$\underline{w}^{(t+1)} = \underline{w}^{(t)} - \tau \sum_{i=1}^{n} \nabla f_i(\underline{w}^{(t)})$$

Today:

@ iteration t. choose $i \in \{1, 2, \ldots, n\}$

$$\underline{w}^{(t+1)} = \underline{w}^{(t)} - \tau \nabla f_{i_t}(\underline{w}^{(t)})$$

- each iteration easier/faster to compute
- need more iterations

How to choose i_k?

A. cyclical ("Incremental Gradient Descent")

$i_t = t \bmod n$

e.g. $n=3$: i_t's = 1, 2, 3, 1, 2, 3, 1, 2, ...

B. random permutations

every n rounds, reshuffle

e.g. $n=3$: i_t's = 1, 3, 2, 3, 1, 2, 2, 1, 3, ...

C. choose i_t uniformly at random

"stochastic gradient descent"

$i_t \sim \text{unif}(1, \ldots, n)$

e.g. $n=3$: i_t's = 1, 3, 3, 2, 3, 1, 2, 2, 2, ...

note: expected value $\mathbb{E}[\nabla f_{i_t}(w)] = \nabla f(\underline{w})$

Ex: $f(\underline{w}) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \langle \underline{x}_i, \underline{w} \rangle)^2 + \lambda \|\underline{w}\|_2^2$

$f_i(w) = (y_i - \langle \underline{x}_i, \underline{w} \rangle)^2 + \lambda \|\underline{w}\|_2^2$

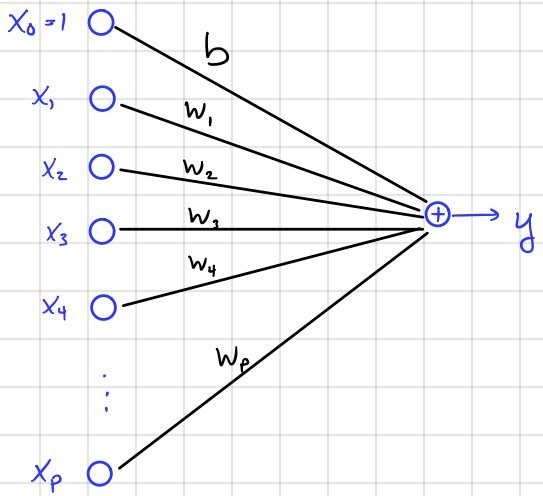
check: $\frac{1}{n}\sum_{i=1}^{n} f_i(\underline{w}) = f(\underline{w})$

$\nabla f_i(\underline{w}) = -2(y_i - \langle \underline{x}_i, \underline{w} \rangle)\underline{x}_i + 2\lambda \underline{w}$

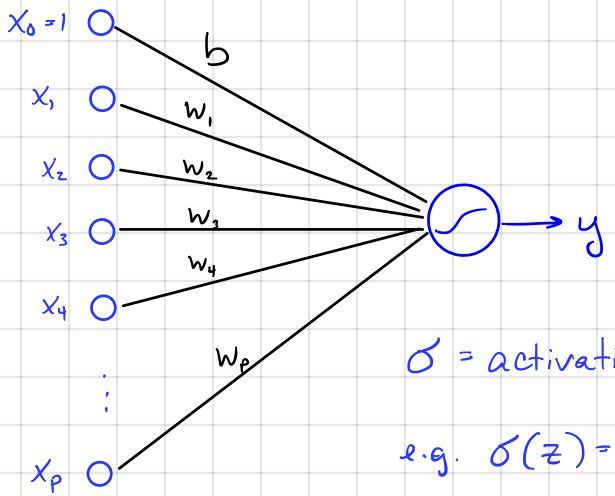
SGD: $\underline{w}^{(t+1)} = \underline{w}^{(t)} + 2\tau(y_{i_t} - \langle \underline{x}_{i_t}, \underline{w}^{(t)} \rangle)\underline{x}_{i_t} - 2\tau\lambda \underline{w}^{(t)}$

A Simple Neural Network

$y = \underline{x}^T \underline{w} + b$



$y = \sigma(\underline{x}^T \underline{w} + b)$



σ = activation function

e.g. $\sigma(z) = \max(0, z)$ = ReLU

$\sigma(z) = \frac{1}{1+e^{-z}}$ = logistic

$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \tanh(z)$

for $\hat{y} = \sigma(\underline{x}^T \underline{w} + b)$ and $\sigma(z) = \frac{1}{1+e^{-z}}$, how do we learn weights?

loss $f(\underline{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sigma(\underbrace{\langle \underline{x}_i, \underline{w} \rangle + b}_{z_i}) \right)^2$$

$\hat{y}_i = \sigma(\underline{x}_i^T \underline{w} + b) = \sigma(z_i)$

$z_i = \underline{x}_i^T \underline{w} + b$

aside: $\frac{d\sigma}{dz} = \sigma'(z)$

$= \sigma(z)(1 - \sigma(z))$

Learn $\underline{w}$ via Stochastic Gradient Descent!

@ iteration t

- choose i_t uniformly at random
- set $\underline{w}^{(t+1)} = \underline{w}^{(t)} - \tau \nabla f_{i_t}(\underline{w}^{(t)})$

$b^{(t+1)} = b^{(t)} - \tau \nabla f_{i_t}(b^{(t)})$

What is $\nabla f_i(w^{(t)})$?

$$\frac{df_i}{dw_j}\Big|_{\underline{w}^{(t)}} = \frac{df_i}{d\hat{y}_i} \cdot \frac{d\hat{y}_i}{dz_i} \cdot \frac{dz_i}{dw_j}\Big|_{\underline{w}^{(t)}}$$

$$= 2(\hat{y}_i - y_i) \cdot \sigma'(z_i) \cdot x_{ij} \Big|_{\underline{w}^{(t)}}$$

$$= 2(\hat{y}_i - y_i) \cdot \sigma(z_i)(1 - \sigma(z_i)) \, x_{ij} \Big|_{\underline{w}^{(t)}}$$

$$= \underbrace{2(\hat{y}_i - y_i)\hat{y}_i(1 - \hat{y}_i)}_{\text{scalar, independent of } j} x_{ij}$$

call $\delta_i = \delta_i(\underline{w}^{(t)})$

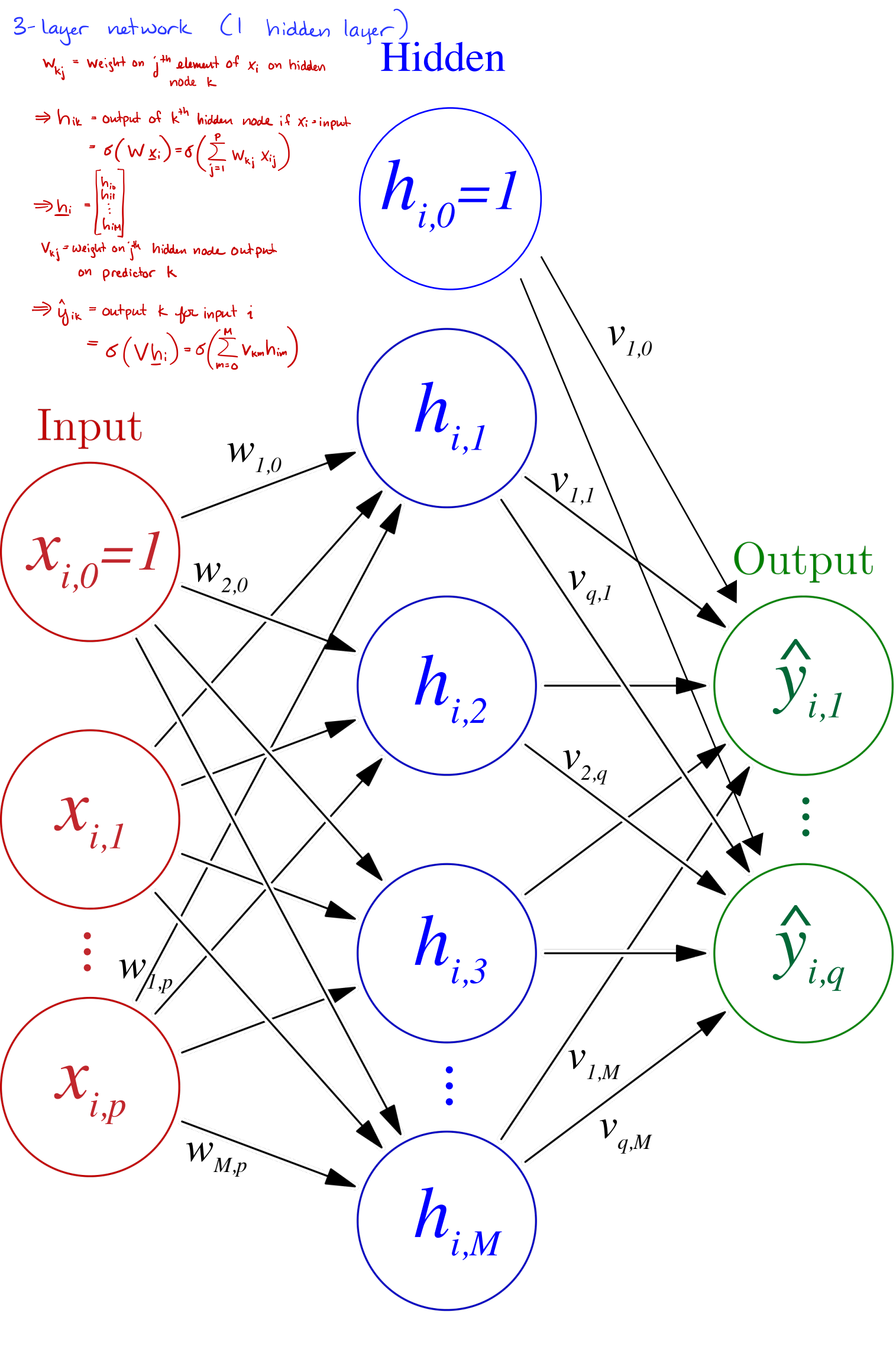
$\Rightarrow \nabla f_i(\underline{w}^{(t)}) = \delta_i \underline{x}_i$

$\Rightarrow$ SGD: $\underline{w}^{(t+1)} = \underline{w}^{(t)} - \tau \delta_{i_t} \underline{x}_{i_t}$, $b^{(t+1)} = b^{(t)} - \tau \delta_{i_t}$

What is $\nabla_b f_{i_t}(b^{(t)}) = \frac{df_{i_t}}{db}\Big|_{b^{(t)}}$?

$$\frac{df_i}{db}\Big|_{b^{(t)}} = \frac{df_i}{d\hat{y}_i} \cdot \frac{d\hat{y}_i}{dz_i} \cdot \frac{dz_i}{db}\Big|_{b^{(t)}}$$

$$= \delta_i \cdot 1$$


3-layer network (1 hidden layer)
w_{kj} = weight on j^{th} element of x_i on hidden node k
$\Rightarrow h_{ik}$ = output of k^{th} hidden node if x_i = input
$= \sigma(W\underline{x}_i) = \sigma\left(\sum_{j=1}^{p} w_{kj} x_{ij}\right)$
$\Rightarrow \underline{h}_i = \begin{bmatrix} h_{i0} \\ h_{i1} \\ \vdots \\ h_{iM} \end{bmatrix}$
v_{kj} = weight on j^{th} hidden node output on predictor k
$\Rightarrow \hat{y}_{ik}$ = output k for input i
$= \sigma(V\underline{h}_i) = \sigma\left(\sum_{m=0}^{M} v_{km} h_{im}\right)$
Hidden
Input
Output
$h_{i,0}=1$
$h_{i,1}$
$h_{i,2}$
$h_{i,3}$
$h_{i,M}$
$x_{i,0}=1$
$x_{i,1}$
$x_{i,p}$
$\hat{y}_{i,1}$
$\hat{y}_{i,q}$
$w_{1,0}$
$w_{2,0}$
$w_{1,p}$
$w_{M,p}$
$v_{1,0}$
$v_{1,1}$
$v_{q,1}$
$v_{2,q}$
$v_{1,M}$
$v_{q,M}$

SGD for 3-layer network:

$$\hat{y}_{ik} = \sigma(V h_i) = \sigma(V \sigma(W \underline{x}_i)) = \sigma\left(\sum_{m=0}^{M} V_{km}\, \sigma\left[\sum_{j=0}^{P} w_{mj}\, x_{ij}\right]\right)$$

Backpropagation algorithm

- Choose initial weights $W^{(1)}$ and $V^{(1)}$
- for $t = 1, 2, \ldots$
  - choose i_t
  - calculate $\hat{y}_{i_t,k}$ and $h_{i_t,m}$ using $W^{(t)}$ and $V^{(t)}$ (forward pass)
  - update weights for each layer, starting with deepest layer (closest to output) and working back to shallowest

To update $V^{(t)}$:

$$\frac{df_i}{dV_{km}} = \frac{df_i}{d\hat{y}_i} \cdot \frac{d\hat{y}_i}{dV_{k,m}}$$

$$= 2(\hat{y}_{ik} - y_{ik})\,\sigma'\left(V^{(t)}\underline{h}_i\right) h_{i,m}$$

$$= \underbrace{2(\hat{y}_{ik} - y_{ik})\,\hat{y}_{i,k}(1-\hat{y}_{i,k})}_{=\delta_{i,k}}\, h_{i,m}$$

$$= \delta_{ik}\, h_{i,m}$$

$$\Longrightarrow V_{k,m}^{(t+1)} = V_{k,m}^{(t)} - 2\tau\,\delta_{i_t,k}\,h_{i_t,m}$$

$$\Longrightarrow \underline{V}_k^{(t+1)} = \underline{V}_k^{(t)} - 2\tau\,\delta_{i_t,k}\,\underline{h}_{i_t} \quad \text{where } \underline{V}_k = \left(k^{th} \text{ row of } V\right)^T = \text{vector of weights determining } k^{th} \text{ predictor}$$

$$\Longrightarrow V^{(t+1)} = V^{(t)} - 2\tau\,\underline{h}_{i_t}\,\underline{\delta}_{i_t}^T \quad \text{where } \underline{\delta}_i^T = \left[\delta_{i,0}\ \ \delta_{i,1}\ \cdots\ \delta_{i,K}\right]$$

To update $W^{(t)}$:

$$\frac{df_i}{dw_{m,j}} = \sum_{k=1}^{q} \frac{df_i}{d\hat{y}_{ik}} \cdot \frac{d\hat{y}_{ik}}{dh_{im}} \cdot \frac{dh_{im}}{dw_{m,j}}$$

$$= 2\sum_{k=1}^{q} (\hat{y}_{ik} - y_{ik})\, \sigma'\left((\underline{v}_k^{(t)})^T \underline{h}_i\right) v_{k,m} \cdot \sigma'\left((\underline{w}_m^{(t)})^T \underline{x}_i\right) x_{ij}$$

$$= \sum_{k=1}^{q} \underbrace{2(\hat{y}_{ik} - y_{ik})\hat{y}_{ik}(1-\hat{y}_{ik})}_{= \delta_{i,k}} v_{k,m}\, h_{im}(1-h_{im})\, x_{ij}$$

$$= \sum_{k=1}^{q} \underbrace{\delta_{ik} v_{k,m} h_{i,m}(1-h_{i,m})}_{=: \gamma_{i,m}} x_{ij}$$

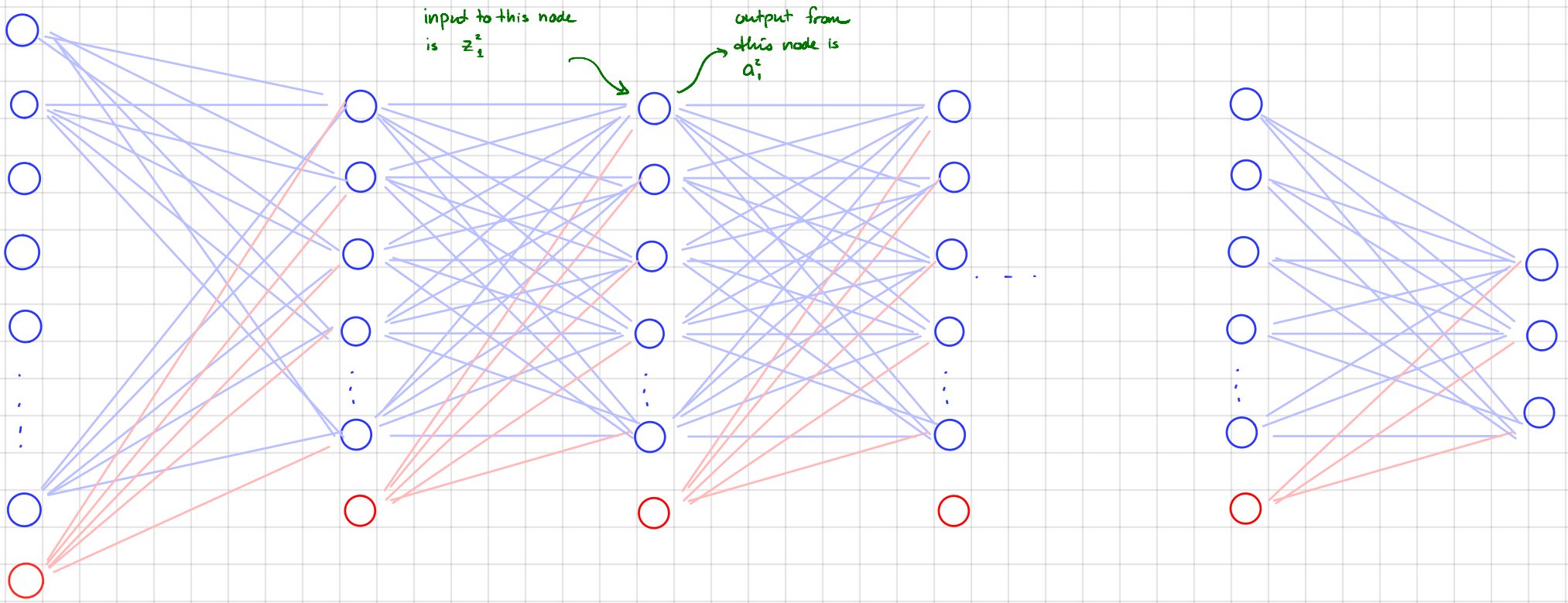
$$= \gamma_{i,m}\, x_{ij}$$

$$\Rightarrow w_{m,j}^{(t+1)} = w_{m,j}^{(t)} - \tau\, \gamma_{i_t,m}\, x_{i_t,j}$$

$$\Rightarrow \underline{w}_m^{(t+1)} = \underline{w}_m^{(t)} - \tau \gamma_{i_t,m}\, \underline{x}_{i_t} \quad \text{where } \underline{w}_m = (m\text{-th row of } W)^T = \text{weights going into } m^{th} \text{ hidden node.}$$

$$\Rightarrow W^{(t+1)} = W^{(t)} - \tau\, \underline{x}_{i_t} \underline{\gamma}_{i_t}^T \quad \text{where } \underline{\gamma}_{i_t}^T = [\gamma_{i_t,0}, \ldots, \gamma_{i_t,M}]$$

More generally



$X = a^0$ $\quad W^1, b^1$ $\quad W^2, b^2$ $\quad W^3, b^3$ $\quad \dots$ $\quad W^L, b^L$

$z^1 = W^1 a^0 + b^1$ $\quad z^2 = W^2 a^1 + b^2$ $\quad z^3 = W^3 a^2 + b^3$ $\quad z^L = W^L a^{L-1} + b^L$

$a^1 = \sigma(z^1)$ $\quad a^2 = \sigma(z^2)$ $\quad a^3 = \sigma(z^3)$ $\quad a^L = \sigma(z^L) = \hat{y}$

$W^l \sim n^l \times n^{l-1}$, $S^l \sim n^l \times 1$, $a^l \sim n^l \times 1$, $b^l \sim n^l \times 1$

z^l is input to nodes in layer l, a^l is output of those nodes

$W^l_{j,k}$ = weight in layer l, applied to layer $l-1$ output (a^{l-1}_k), feeding into next layer (z^l_j)

To update W^l, need to compute $\nabla f(W^l)$

$$\frac{df}{dW^l_{j,k}} = \frac{df}{dz^l_j} \cdot \frac{dz^l_j}{dW^l_{j,k}} = \delta^l_j \cdot a^{l-1}_k \implies \nabla f(W^l) = \underline{\delta}^l \left(\underline{a}^{l-1}\right)^T$$

element-wise multiplication

$$\delta^l_j := \frac{df}{dz^l_j} = \begin{cases} \frac{df}{da^l_j} \cdot \frac{da^l_j}{dz^l_j} = \left[\left(W^{l+1}\right)^T \delta^{l+1}\right]_j \cdot \sigma'(z^l_j) & \text{for } l < L \\ \frac{df}{a^l_j}\sigma'(z^l_j) & \text{for } l = L \end{cases} \implies \underline{\delta}^l = \begin{cases} \left(\left(W^{l+1}\right)^T \underline{\delta}^{l+1}\right) \odot \sigma'(\underline{z}^l) & l < L \\ \nabla f(\underline{a}^L) \odot \sigma'(\underline{z}^l) & l = L \end{cases}$$

$$\frac{df}{da^l_j} = \sum_k \frac{df}{dz^{l+1}_k} \frac{dz^{l+1}_k}{da^l_j} = \sum_k \delta^{l+1}_k W^{l+1}_{kj} = \left[\left(W^{l+1}\right)^T \delta^{l+1}\right]_j$$

$$\frac{da^l_j}{dz^l_j} = \sigma'(z^l_j)$$

$$\frac{dz^l_j}{dW^l_{j,k}} = a^{l-1}_k \quad \text{because} \quad z^l_j = \left(W^l a^{l-1} + b^l\right)_j = \sum_k W^l_{jk} a^{l-1}_k + b^l_j$$

$$\frac{df}{db^l_j} = \frac{df}{dz^l_j} \cdot \frac{dz^l_j}{db^l_j} = \delta^l_j \cdot 1 \implies \nabla f(b^l) = \delta^l_j$$

Backpropagation Algorithm

\# forward pass:

$a^0 = \sigma(\underline{x}_{i_t})$

for $l = 1, 2, \ldots, L$

$\underline{z}^l = W^l \underline{a}^{l-1} + \underline{b}^l$

$\underline{a}^l = \sigma(z^l)$

end

$\underline{\delta}^L = \nabla f(a^L) \cdot \sigma'(z^L)$

\# backprop

for $l = L-1, L-2, \ldots, 1$

$\underline{\delta}^l = \left[(W^{l+1})^T \delta^{l+1} \right] \odot \sigma'(\underline{z}^l)$

$\nabla f(W^l) = \underline{\delta}^l (\underline{a}^{l-1})^T$

$\nabla f(b^l) = \underline{\delta}^l$

$W^l \leftarrow W^l - \tau \nabla f(W^l)$

$b^l \leftarrow b^l - \tau \nabla f(b^l)$

end