MATH 565: Lecture 26 (04/16/2026)

Today: * Loss function to w derivatives
* CGs with vector variables

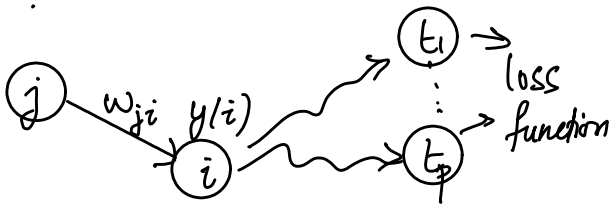
Loss Function to Weight Derivatives

Recall: $S(s,t) = \frac{\partial y(t)}{\partial y(s)}$, node-to-node derivative and the DP to compute $S(s,t)$ using back propagation.

The output node values ($y(t)$ or $y(t_i)$ for $i=1,\dots,p$) are used to compute loss function J. And we want to compute $\nabla_w J$ (or $\frac{\partial J}{\partial w}$) for edge weights w_{ji}, $(j,i)\in E$.

We apply the chain rule again!

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y(i)} \cdot \frac{\partial y(i)}{\partial w_{ji}}$$

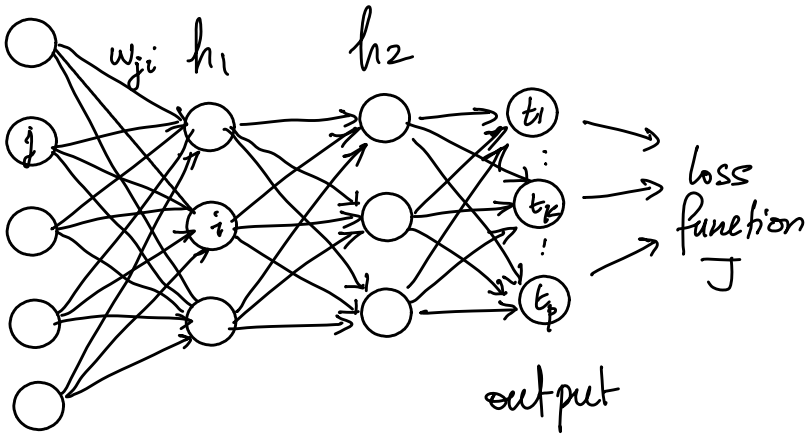


If there are p output nodes $t_1,\dots,t_p$, then

$$\frac{\partial J}{\partial w_{ji}} = \sum_{k=1}^{p} \left[\frac{\partial J}{\partial y(t_k)} \cdot \frac{\partial y(t_k)}{\partial y(i)}\right] \cdot \frac{\partial y(i)}{\partial w_{ji}}$$

We can modify the previous DP algorithm to compute $\frac{\partial L}{\partial w_{ji}}$ instead. Instead of $S(j,t)$, we use

$$\Delta(i) = \frac{\partial J}{\partial y(i)}$$ and we back propagate the loss derivatives all the way to the input nodes.



We modify the original DP algorithm to the present context.

$$\Delta(t_k) = \frac{\partial J}{\partial y(t_k)}, \quad k = 1, \ldots, p \qquad \text{(initialization)}$$

do

select unprocessed $i \in \mathcal{N}$ for which $\Delta(j)$ is known $\forall\, j \in A(i)$;

$$\Delta(i) \leftarrow \sum_{j \in A(i)}{}' \Delta(j) \cdot z(i,j)$$

while $\exists\, i \in \mathcal{N}$ unprocessed

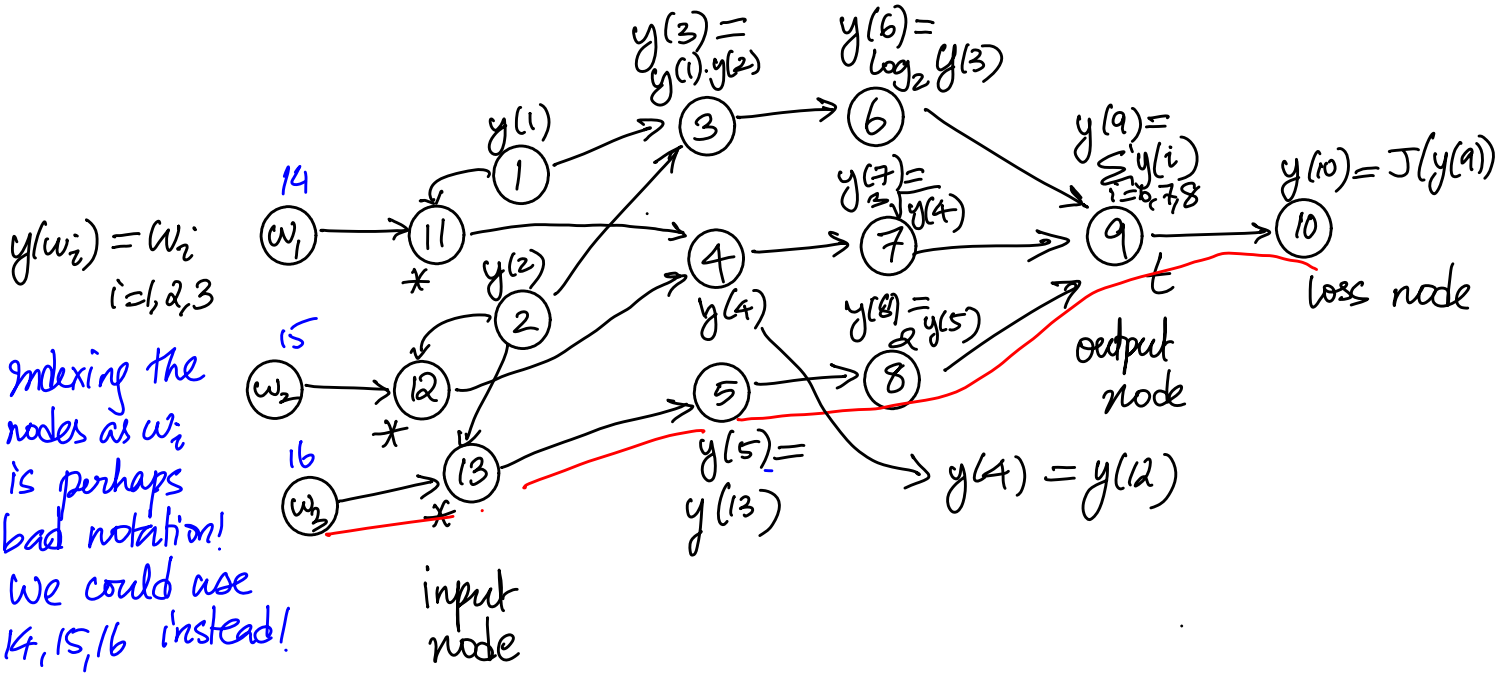
for $(j,i) \in \mathcal{E}$ with edge weight w_{ji}

$$\frac{\partial J}{\partial w_{ji}} \leftarrow \Delta(i) \cdot \frac{\partial y(i)}{\partial w_{ji}}$$

end

Another Approach: Stick with node-to-node derivatives, and add extra nodes to capture edge weights and the loss function. This is a standard network transformation; the other transformation of replacing a node with an edge is more standard in network flows.

Illustration on Example



$$y(11) = w_1 \cdot y(1) \qquad y(12) = w_2 \cdot y(2). \qquad y(13) = w_3 \cdot y(2)$$

$y(w_1)$.

After the network is transformed in this manner, we can run the node-to-node back propagation DP.

Now, we can run standard gradient descent (GD):

$$\bar{w} \leftarrow \bar{w} - \alpha \nabla_{\bar{w}} J \qquad \nabla_{\bar{w}} J = \frac{\partial J}{\partial \bar{w}}$$

α: learning rate

Illustration

Let the loss function be $y(10) = J(y(9)) = \ln(y(9))$.

$$\Rightarrow \frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial y(w_3)} = \frac{\partial y(10)}{\partial y(9)} \cdot S(5,9) \cdot \frac{\partial y(5)}{\partial y(13)} \cdot \frac{\partial y(13)}{\partial y(w_3)}$$

$\frac{\partial y(10)}{\partial y(9)} = \frac{1}{y(9)}$, $S(5,9) = 2^{y(5)} \ln 2$, $\frac{\partial y(5)}{\partial y(13)} = 1$, $\frac{\partial y(13)}{\partial y(w_3)} = y(2)$

$$= \frac{1}{y(9)} \cdot 2^{y(5)} \ln 2 \cdot y(2),$$ which can be computed using numerical differentiation.

Computational Graphs with Vector Variables

We had seen $z(i,j) = \frac{\partial y(j)}{\partial y(i)}$ and $S(s,t) = \frac{\partial y(t)}{\partial y(s)}$. In more general CGs, $\bar{y}(i) \in \mathbb{R}^d$ for some $d > 1$. The derivative is now replaced by the Jacobian.

Def For $\bar{f}: \mathbb{R}^n \to \mathbb{R}^m$, $\bar{f}(\bar{x}) = \begin{bmatrix} f_1(\bar{x}) \\ \vdots \\ f_m(\bar{x}) \end{bmatrix}$ $\bar{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

the Jacobian is

$$\mathcal{D}(\bar{f}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{m \times n} = \begin{bmatrix} (\nabla_{\bar{x}} f_1)^T \\ \vdots \\ (\nabla_{\bar{x}} f_m)^T \end{bmatrix}.$$

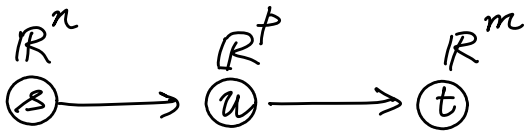
or $\mathcal{D}_{\bar{f}}$ or $\mathcal{D}\bar{f}$

↳ gradients of f_i written as rows.

If we have $g: \mathbb{R}^m \to \mathbb{R}^p$, then

$$\underset{p \times n}{\mathcal{D}(g \circ f)} = \underset{p \times m}{\mathcal{D}(\bar{g})} \cdot \underset{m \times n}{\mathcal{D}(\bar{f})}$$

Let $\bar{y}(t) \in \mathbb{R}^m$, $\bar{y}(s) \in \mathbb{R}^n$. Then $\mathcal{D}(s,t)$ is an $m \times n$ matrix of partial derivatives.

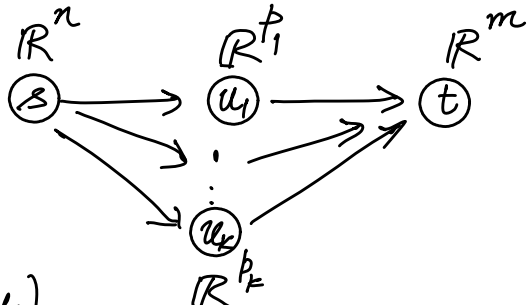


With a simple s-t path, we get

$$\underset{m \times n}{\mathcal{D}(s,t)} = \underset{m \times p}{\mathcal{D}(u,t)} \cdot \underset{p \times n}{\mathcal{D}(s,u)}$$

*! The order is important here — it should honor matrix multiplication!

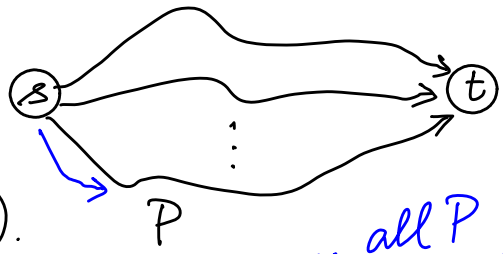
If there are k s-t paths in this fashion, we get



$$\underset{m\times n}{\mathcal{D}(s,t)} = \sum_{i=1}^{k} \underset{m\times p_i}{\mathcal{D}(u_i,t)} \cdot \underset{p_i\times n}{\mathcal{D}(s,u_i)}$$

Here is the result in general.

Let $\mathcal{P}=\{P_i\}$ be the collection of s-t paths. Each path P_i has k_i edges ($k_i \geq 1$).



all P need not have the same k!

For brevity of notation, consider path P with k edges of the form

$$P: \; s = v_0 \to v_1 \to \cdots \to v_k = t. \quad \text{Then}$$

$$\mathcal{D}(s,t) = \sum_{P\in\mathcal{P}} \underset{v_k}{\mathcal{D}(v_{k-1},t)} \cdot \mathcal{D}(v_{k-2},v_{k-1}) \cdots \mathcal{D}(v_1,v_2) \cdot \underset{v_0}{\mathcal{D}(s,v_1)}.$$

Note the order of multiplication: you start with the term closest to s on the rightmost end of the product.

We can generalize the DP algorithms for node-to-node and loss-to-weight derivative calculations to the vector setting.