MATH 565: Lecture 25 (04/14/2026)

Today: * pathwise aggregation lemma
* Dynamic programming approach

Lemma 13 Let $G_c = (N, \mathcal{E})$ be a DACG with node variable $y(i)$ associated with $i \in N$. The local derivative $z(i,j)$ of $(i,j) \in \mathcal{E}$ is defined as

$$z(i,j) = \frac{\partial y(j)}{\partial y(i)}$$

Let $\mathcal{P} \neq \emptyset$ be a set of paths from s to t for $s, t \in N$. Then

$$\frac{\partial y(t)}{\partial y(s)} = \sum_{P \in \mathcal{P}} \prod_{(i,j) \in P} z(i,j).$$

The proof is a direct application of multivariate chain rule of differentiation.

Dynamic Programming (DP) for node-to-node derivatives

DP is used widely in optimization, e.g., Dijkstra's algorithm for shortest path (SP) computation in directed networks.

Aside on Dijkstra's algorithm for SP.

$G = (N, A)$ N: nodes, A: arcs (directed), $c_{ij} \geq 0$: cost for $(i,j) \in A$ for $i, j \in N$.

goal: find SP distances from source node $s \in N$ to all $j \in N$.

Let $d(i)$ be the distance label of $i \in N$.

$d(i)$ = upper bound on SP distance from s to i.

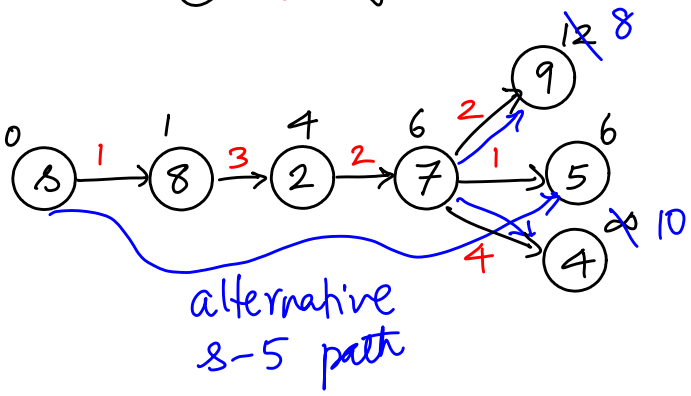
Initialization: $d(i) = +\infty \ \forall i \in N \setminus \{s\}$, $d(s) = 0$.

A key step in Dijkstra's algorithm: UPDATE(i)

```
for each (i,j) ∈ A do
    if d(j) > d(i) + c_ij
        d(j) = d(i) + c_ij ;
    end if
endfor
```

This procedure goes forward from s.

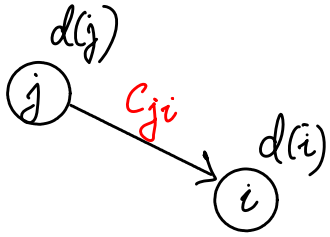
UPDATE(7)



We can also apply Dijkstra backward starting from t.

UPDATE(i)

```
for each (j,i) ∈ A do
    if d(j) > c_ji + d(i)
        d(j) = c_ji + d(i) ;
    end if
endfor
```



$d(i)$ = upper bound for SP distance from i to t.

This backward version is quite similar in structure to our derivative computation...

Back to node-to-node derivatives...

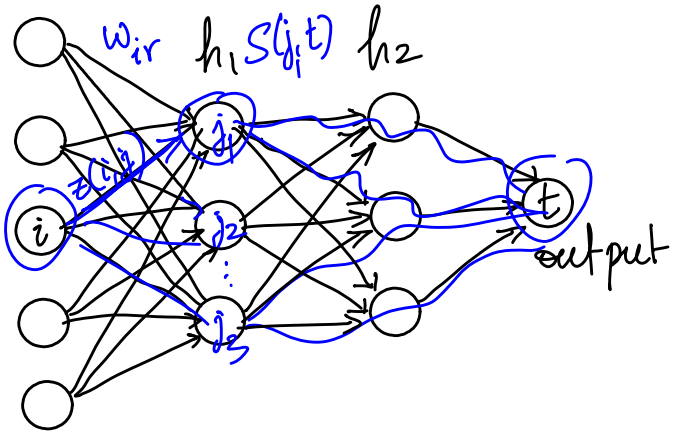
Let $S(s,t) = \frac{\partial y(t)}{\partial y(s)}$

DP update:

$$S(i,t) \leftarrow \sum_{j \in A(i)}' S(j,t) \cdot z(i,j)$$

where $A(i) = \{$ head nodes of outarcs of node $i\}$

$$= \{ j \mid (i,j) \in \mathcal{E} \}$$



We can start by setting

$S(t,t) = \frac{\partial y(t)}{\partial y(t)} = 1$.

(initialization)

$S(t,t)=1$ (initialization)

do

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

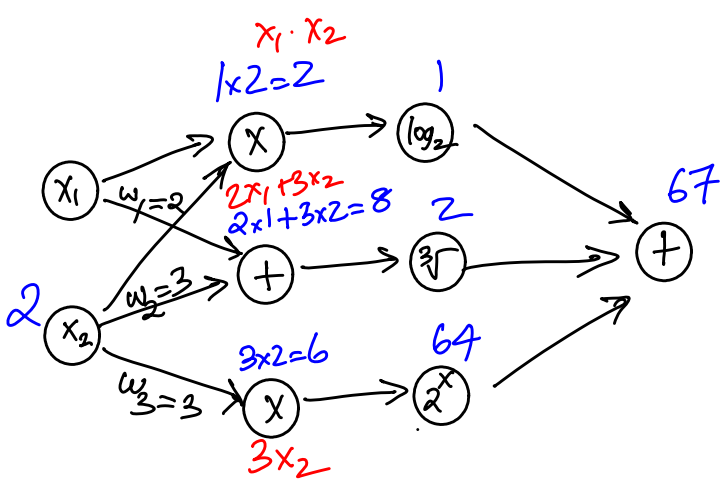
$$S(i,t) \leftarrow \sum_{j \in A(i)}' S(j,t) \cdot z(i,j)$$

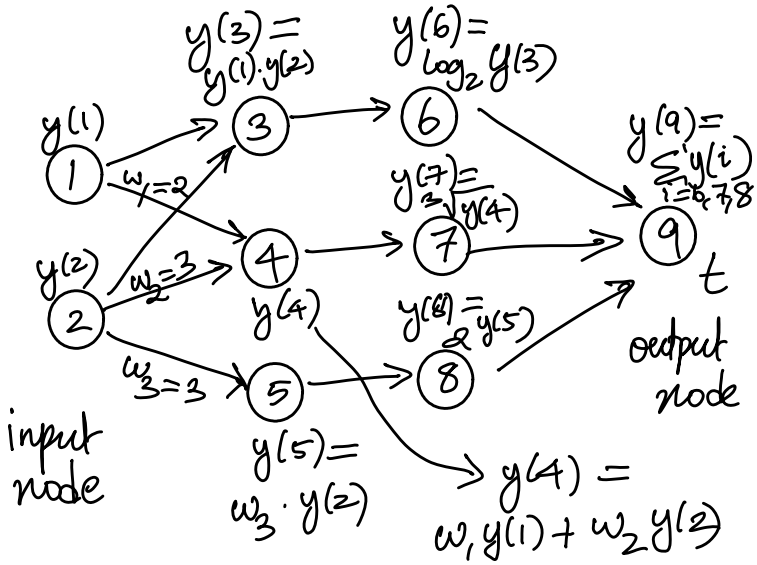
while $\exists\, i \in \mathcal{N}$ unprocessed

Such unprocessed nodes always exist in DACGs.
Node selection is done in a backward direction starting at node t — hence it is called the **backpropagation algorithm**.

Example from Lecture 23

We number the nodes from 1 to 9:





We want to compute $S(1,9)$ and $S(2,9)$.

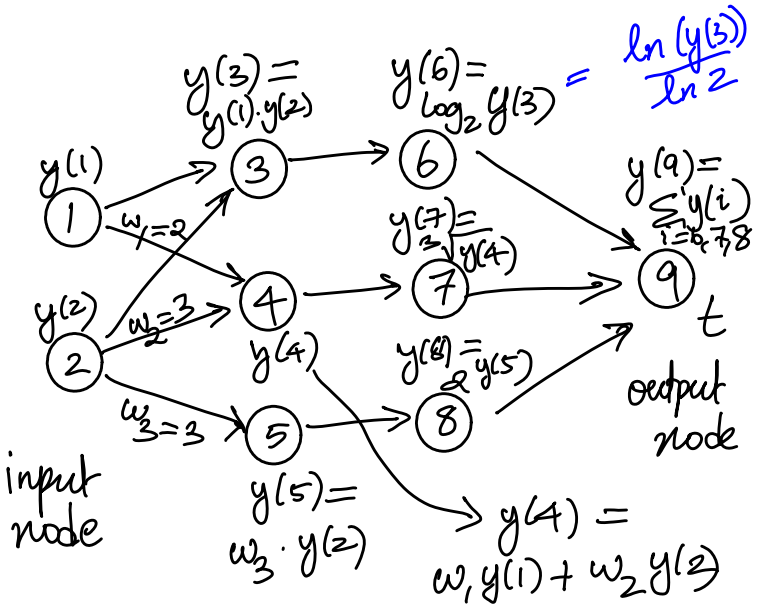
$S(9,9) = 1$ (initialization)

As $y(9) = y(6) + y(7) + y(8)$, we get

$$z(i,9) = \frac{\partial y(9)}{\partial y(i)} = 1, \quad i = 6, 7, 8.$$

$$\Rightarrow S(i,9) = \underbrace{S(9,9)}_{=1} \cdot z(i,9), \quad i = 6, 7, 8$$

$$\Rightarrow S(i,9) = 1, \quad i = 6, 7, 8.$$



$$z(3,6) = \frac{1}{\ln 2}\cdot\frac{1}{y(3)}$$

$$z(4,7) = \frac{1}{3}(y(4))^{-2/3}$$

$$z(5,8) = 2^{y(5)}\cdot\ln 2.$$

$$\Rightarrow S(3,9) = S(6,9)\cdot z(3,6) = 1\cdot z(3,6) = \frac{1}{\ln 2}\cdot\frac{1}{y(3)};$$

$$S(4,9) = S(7,9)\cdot z(4,7) = 1\cdot z(4,7) = \frac{1}{3}(y(4))^{-2/3};$$

$$S(5,9) = S(8,9)\cdot z(5,8) = 1\cdot z(5,8) = 2^{y(5)}\cdot\ln 2.$$

$z(1,3) = y(2)$
$z(1,4) = w_1$
$z(2,3) = y(1)$
$z(2,4) = w_2$
$z(2,5) = w_3$

$$S(1,9) = S(3,9)\cdot z(1,3) + S(4,9)\cdot z(1,4) = \frac{y(2)}{\ln 2\cdot y(3)} + \frac{w_1}{3}(y(4))^{-2/3}$$

$$S(2,9) = S(3,9)\cdot z(2,3) + S(4,9)\cdot z(2,4) + S(5,9)\cdot z(2,5)$$

$$= \frac{y(1)}{\ln 2\cdot y(3)} + \frac{w_2}{3}(y(4))^{-2/3} + w_3\ln 2\cdot 2^{y(5)}$$

We can double check that

$$S(1,9) = \frac{\partial f_G}{\partial x_1} \text{ and } S(2,9) = \frac{\partial f_G}{\partial x_2},$$

where $f_G = \log_2(x_1\cdot x_2) + \sqrt[3]{2x_1+3x_2} + 2^{3x_2}$ is the global function fitted...

But in general, we do NOT want to symbolically compute these expressions (in general). Instead, we numerically compute $y(i)$ $\forall$ $i \in N$ for each sample (x_i, y_i). Hence, the procedure is called numerical differentiation.