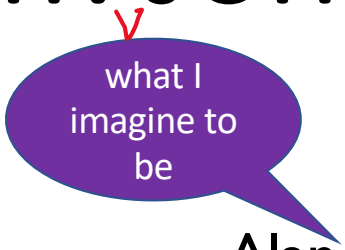
AutoDiff: John Gilbert Style



Alan Edelman

(Ekin Akyürek, Yuyang Wang)

June 3rd, 2023

CSAIL

MIT MATHEMATICS

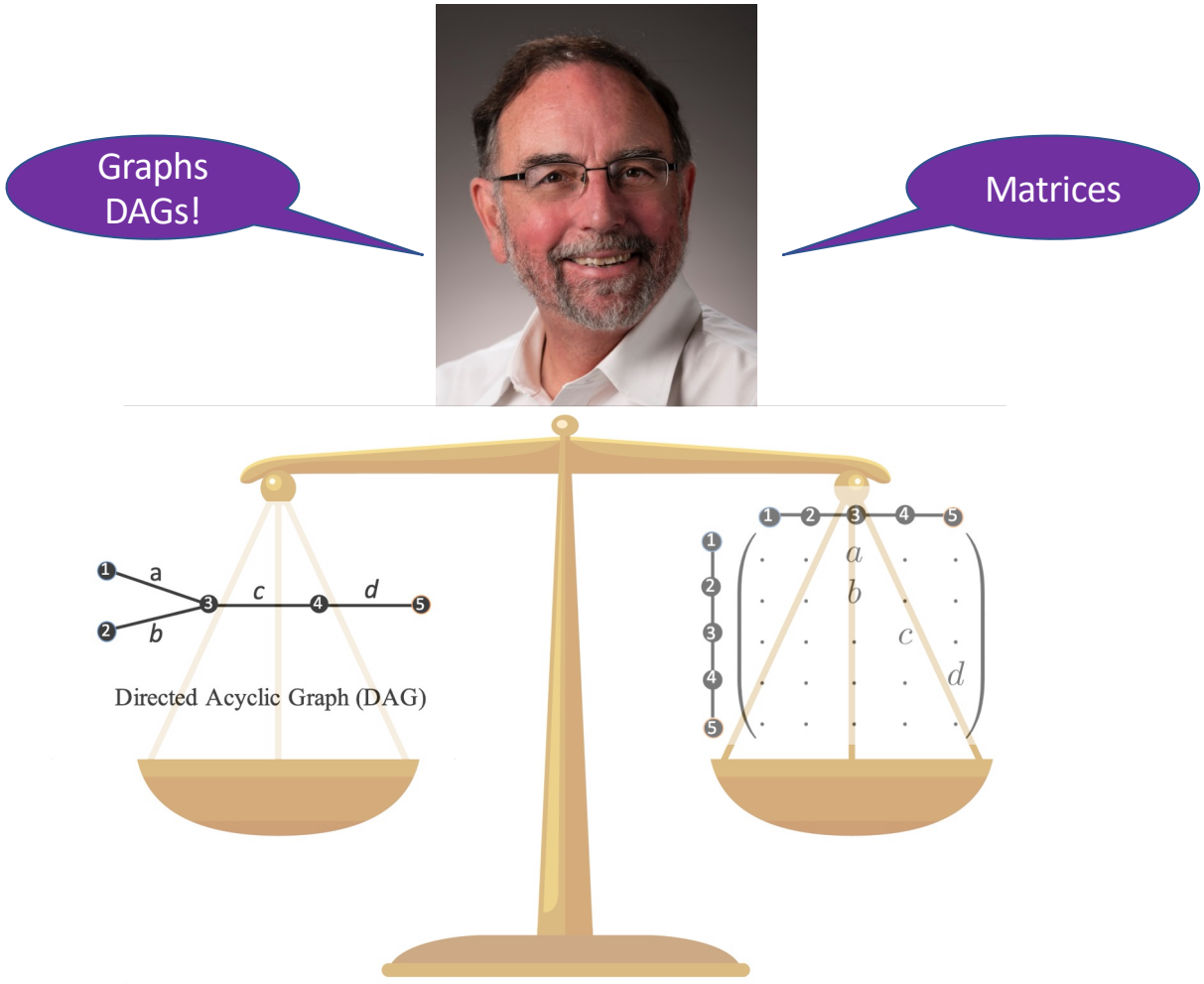
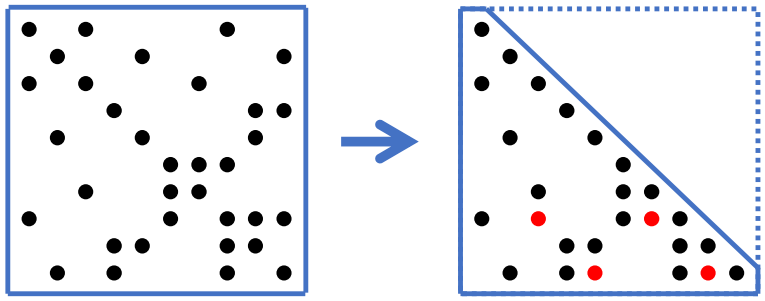


image: Image by brgfx on Freepik

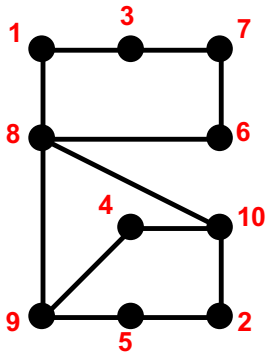
Graphs and Sparse Matrices: Cholesky factorization



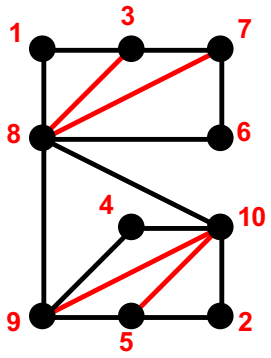
Fill: new nonzeros in factor

How many of you haven't seen this slide?

made for 18.337/2004 Sparse days I?



G(A)



$G^+(A)$
[chordal]

Symmetric Gaussian elimination:

for j = 1 to n

add edges between j's higher-numbered neighbors

John Gilbert: The interplay between graphs and matrices

JRG70 Agenda : Saturday Technical Program

Time	Topic	Speaker
8:15-9:00am	Light breakfast and opening remarks	
9:00-9:30am	Autodiff: John Gilbert Style	Alan Edelman
9:30-10:00am	Graphs, Matrices and John Gilbert	Alex Pothen
10:00-10:30am	A Matrix Laboratory	Cleve Moler
10:30-11:00am	Row replicated block Cimmino	Iain Duff
11:00-11:30am	The role of sparse factorization in scientific computing	Sherry Li
11:30-12noon	Towards Two to Five Truths Revealed in Non-Negative Matrix Factorizations	John M. Conroy
12:00-1:30pm	Lunch and mingling	
1:30-2:00pm	Fun with Big Models: Beyond Graph and Matrix	Shang-Hua Teng
2:00-2:30pm	Building sparse matrix and graph abstractions with multiple dispatch in Julia	Viral Shah
2:30-3:00pm	Sparse matrices are all you need(?)	Aydin Buluc
3:00-3:30pm	LARC: Linear Algebra via Recursive Compression	Jenny Zito
3:30-4:00pm	John's Left-Looking Sparse LU: Elegance and Moore's Law Zeitgeist	Sivan Toledo
4:00-5:00pm	Open microphone and remarks by JRG	John Gilbert and others

Graphs and Matrices

- In the end it is a notation
  - From a deep mathematical view, there is no distinction
- But the interplay is amazing in how we understand

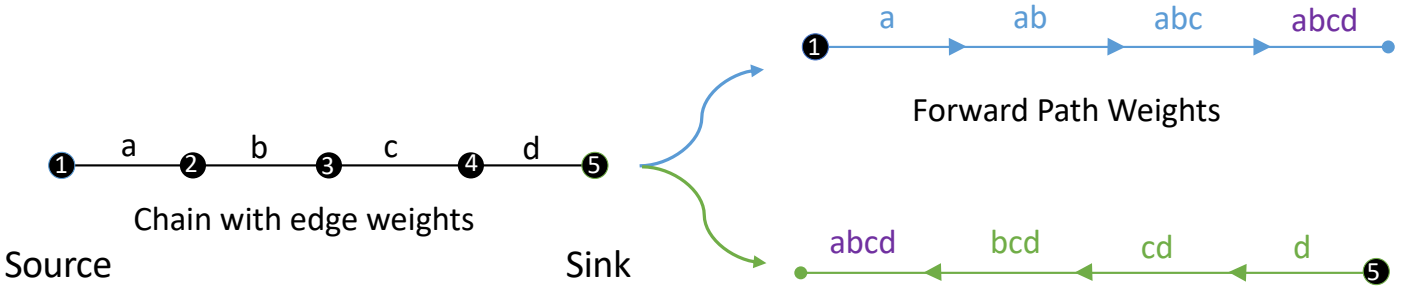
Automatic Differentiation
Scalar Example: sin(1/x)

- Forward Mode: $(-1/x^2)*\cos(1/x)$ (left to right)
- Reverse Mode: $\cos(1/x)*(-1/x^2)$. (left to right)

Everyone says that automatic differentiation is really just the chain rule …

I say, it's just **linear algebra** **and** **graph theory**!

Path Weights



$$L^T = \begin{pmatrix} . & a & . & . & . \\ . & . & b & . & . \\ . & . & . & c & . \\ . & . & . & . & d \\ . & . & . & . & . \end{pmatrix}$$

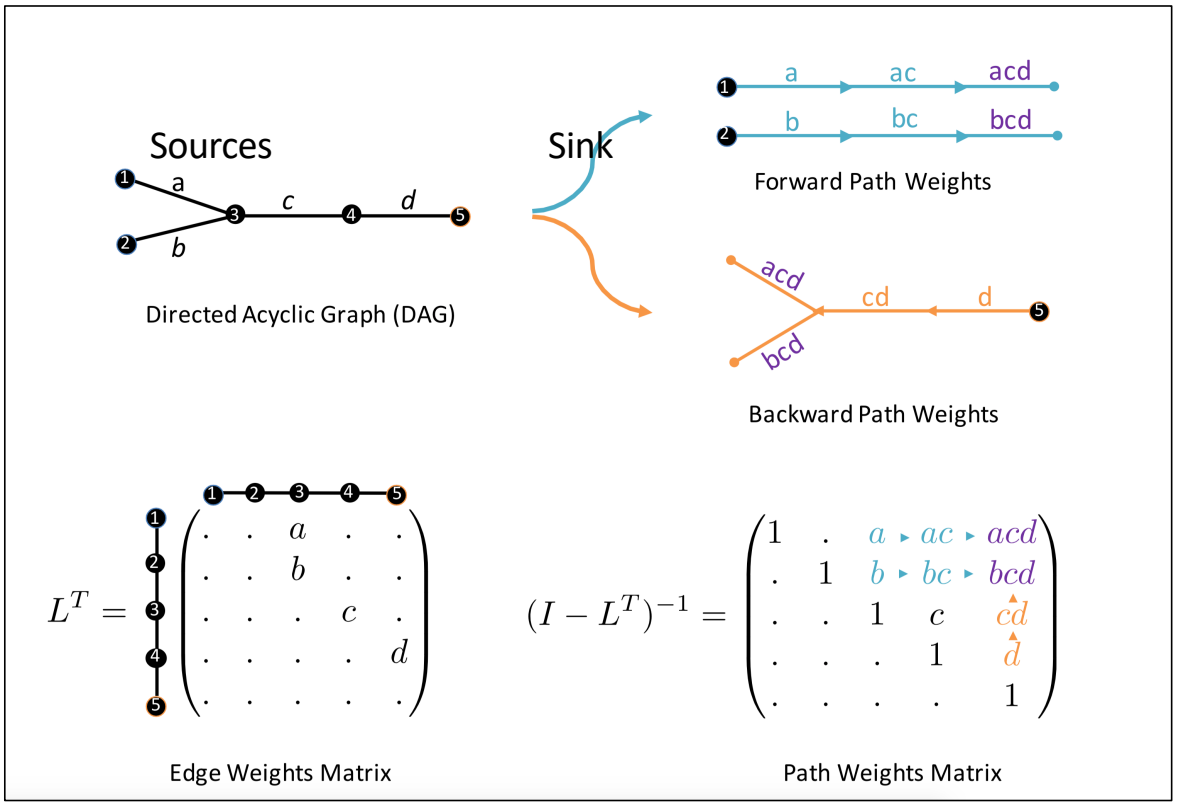
Edge Weights Matrix

$$(I - L^T)^{-1} = \begin{pmatrix} 1 & a & ab & abc & abcd \\ . & 1 & b & bc & bcd \\ . & . & 1 & c & cd \\ . & . & . & 1 & d \\ . & . & . & . & 1 \end{pmatrix}$$

Path Weights Matrix

b,bc black, bcd orange

Path Weights



$$\text{path weights} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T}_{\text{sources}} (I - L^T)^{-1} \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}}_{\text{sink}},$$

from sources to sink

How to Compute

Forward Substitution:

$$\begin{pmatrix}0\\0\\0\\0\\1\end{pmatrix}^T \left[(I-L)^{-1} \begin{pmatrix}1 & 0\\0 & 1\\0 & 0\\0 & 0\\0 & 0\end{pmatrix} \right]$$

Back Substitution:

$$\begin{pmatrix}1 & 0\\0 & 1\\0 & 0\\0 & 0\\0 & 0\end{pmatrix}^T \left[(I-L^T)^{-1} \begin{pmatrix}0\\0\\0\\0\\1\end{pmatrix} \right]$$

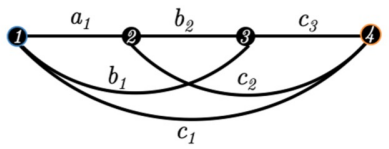
3.1.2. Generalizing "Forward and Back" to a Catalan number of possibilities. Continuing with the same L matrix from Section 3.1, we can begin to understand all of the possibilities including the forward approach, the approach, the mixed-modes approaches, and even more possibilities:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T (I - L^T)^{-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} =$$

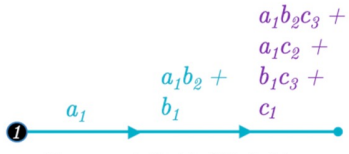
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T \begin{pmatrix} 1 & . & a & . & . \\ . & 1 & . & . & . \\ . & . & 1 & . & . \\ . & . & . & 1 & . \\ . & . & . & . & 1 \end{pmatrix} \begin{pmatrix} 1 & . & . & . & . \\ . & 1 & b & . & . \\ . & . & 1 & . & . \\ . & . & . & 1 & . \\ . & . & . & . & 1 \end{pmatrix} \begin{pmatrix} 1 & . & . & . & . \\ . & 1 & . & . & . \\ . & . & 1 & c & . \\ . & . & . & 1 & . \\ . & . & . & . & 1 \end{pmatrix} \begin{pmatrix} 1 & . & . & . & . \\ . & 1 & . & . & . \\ . & . & 1 & . & . \\ . & . & . & 1 & d \\ . & . & . & . & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

It is well known [11], that there are a Catalan number, $C_5 = 42$, ways to parenthesize the above expression. One of the 42 parenthesizations evaluates left to right; this is forward substitution which computes the graph weights forward. Another evaluates from right to left is backward substitution. There are three other "mixed-modes" [9] which combine forward and backward methods. The remaining 37 methods require matrix-matrix multiplication as a first step. We encourage the reader to work out some of these on the graph. Partial products correspond to working through subgraphs. Perhaps readers might find cases where working from the middle outward

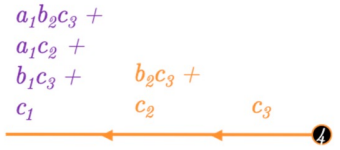
Example: The Complete DAG



Complete DAG



Forward Path Weights



Backward Path Weights

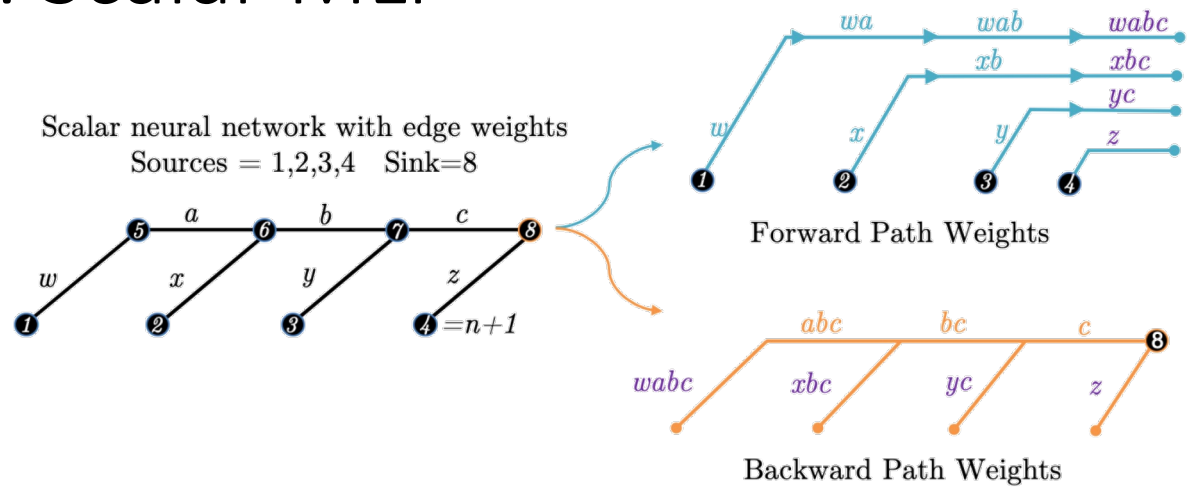
$$L^T = \begin{pmatrix} . & a_1 & b_1 & c_1 \\ . & . & b_2 & c_2 \\ . & . & . & c_3 \\ . & . & . & . \end{pmatrix}, \quad (I - L^T)^{-1} = \begin{pmatrix} 1 & a_1 & a_1b_2 + b_1 & \{a_1b_2c_3 + a_1c_2 \\ & & & +b_1c_3 + c_1\} \\ . & 1 & b_2 & b_2c_3 + c_2 \\ . & . & 1 & c_3 \\ . & . & . & 1 \end{pmatrix}$$

Edge Weights Matrix

Path Weights Matrix

Note the symmetry between forward and backward in this case.

Example: Scalar MLP



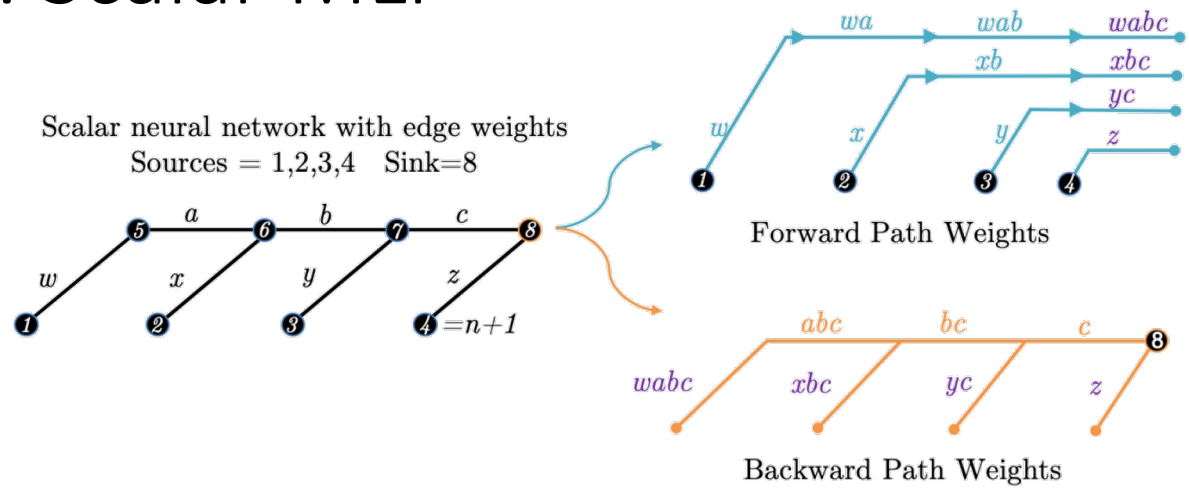
$L^T =$

Edge Weights Matrix

$(I - L^T)^{-1} =$

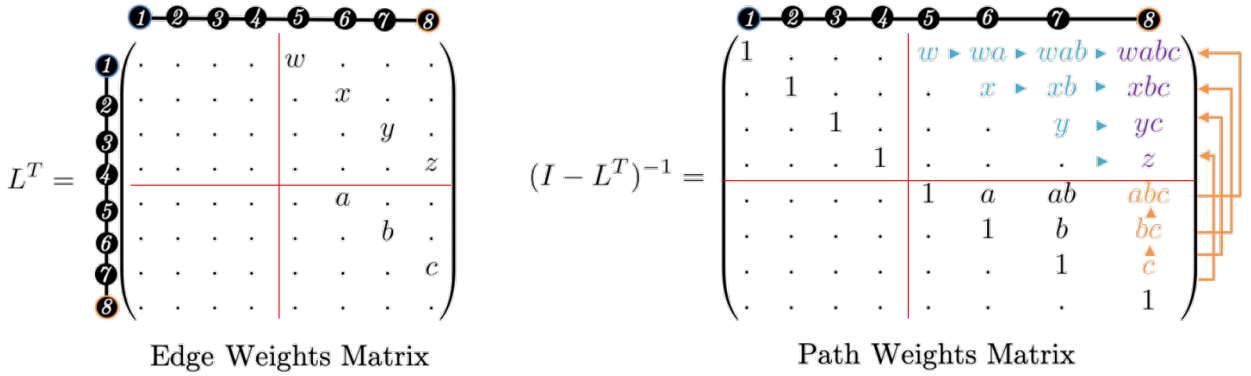
Path Weights Matrix

Example: Scalar MLP



"Natural Block Decomposition"

Source — Internal
Internal—Internal



Edge Elimination

eliminate edge 3-4

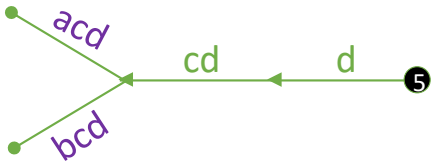
Directed Acyclic Graph

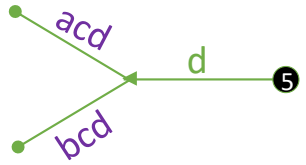
Edge Weights Matrix

$$L^T = \begin{pmatrix} \cdot & \cdot & a & \cdot & \cdot \\ \cdot & \cdot & b & \cdot & \cdot \\ \cdot & \cdot & \cdot & c & \cdot \\ \cdot & \cdot & \cdot & \cdot & d \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} + \begin{pmatrix} a \\ b \\ \cdot \\ c \\ \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ c \\ \cdot \end{pmatrix}^T = \begin{pmatrix} \cdot & \cdot & ac & \cdot \\ \cdot & \cdot & bc & \cdot \\ \cdot & \cdot & \cdot & d \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Forward Path Weights

Backward Path Weights



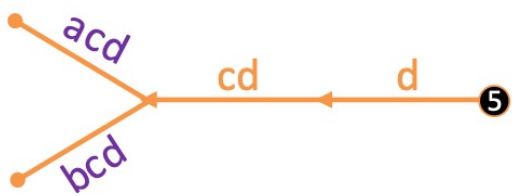


Path Weights Matrix

$$(I - L^T)^{-1} = \begin{pmatrix} 1 & . & a & ac & acd \\ . & 1 & b & bc & bcd \\ . & . & 1 & c & cd \\ . & . & . & 1 & d \\ . & . & . & . & 1 \end{pmatrix} = \begin{pmatrix} 1 & . & ac & acd \\ . & 1 & bc & bcd \\ . & . & 1 & d \\ . & . & . & 1 \end{pmatrix}$$

Edge Addition

Add an edge e



$$\text{updated path weights} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T}_{\text{sources}} \underbrace{\begin{pmatrix} (I - L^T)^{-1} & . \\ . & 1 \end{pmatrix} \begin{pmatrix} 1 & . & . & . & . & . \\ . & 1 & . & . & . & . \\ . & . & 1 & . & . & . \\ . & . & . & 1 & . & . \\ . & . & . & . & 1 & e \\ . & . & . & . & . & 1 \end{pmatrix}}_{\text{updated path weights matrix}} \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}}_{\text{sink}}.$$

$$\text{path weights} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T}_{\text{sources}} \underbrace{(I - L^T)^{-1}}_{\text{path weights matrix}} \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}}_{\text{sink}}$$

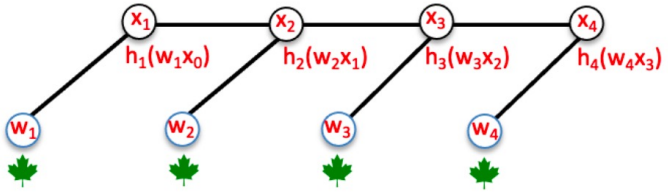
$$\text{updated path weights} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}^T}_{\text{sources}} (I - L^T)^{-1} \underbrace{\begin{pmatrix} . \\ . \\ . \\ . \\ e \end{pmatrix}}_{\text{sink}}.$$

Putting it all together

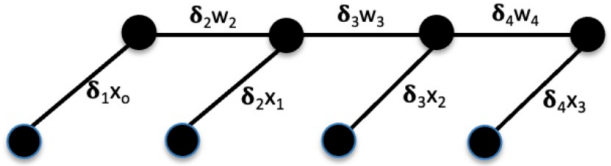
```
∇J = M' * (ImL' \ g)
```

Wait: where are the derivatives?

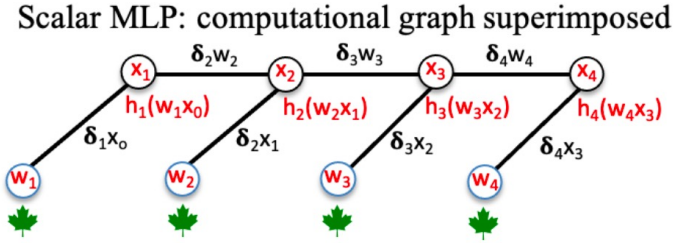
Scalar MLP: computational graph



Scalar MLP (edge weight) derivative graph



Scalar MLP: computational graph superimposed



Vertices:Computational Graph

Edges: ”One step” derivatives

Matrix Operators

	Symbol	Definition	Dense Representation	
Kronecker Product of A, B	$A \otimes B$	$X \mapsto BXA^T$	$A \otimes B$	$m_1 n_1 \times mn$
Left Multiplication by B	B_L	$X \mapsto BX$	$I \otimes B$	$m_1 n \times mn$
Right Multiplication by A	A_R	$X \mapsto XA$	$A^T \otimes I$	$mn_1 \times mn$
Hadamard Product with M	M_H	$X \mapsto M. * X$	$\text{diag}(\text{vec}(M))$	$mn \times mn$
Matrix inner product with G	$G^T \bullet$	$X \mapsto \text{trace}(G^T X)$	$\text{vec}(G)^T$	$1 \times mn$

(We overload $A \otimes B$ to be both the operator and the matrix.)

Matrix Adjoints

	Symbol	Definition	Dense Representation	
Kronecker Product of A, B	$A \otimes B$	$X \mapsto BXA^T$	$A \otimes B$	$m_1 n_1 \times mn$
Left Multiplication by B	B_L	$X \mapsto BX$	$I \otimes B$	$m_1 n \times mn$
Right Multiplication by A	A_R	$X \mapsto XA$	$A^T \otimes I$	$mn_1 \times mn$
Hadamard Product with M	M_H	$X \mapsto M. * X$	$\text{diag}(\text{vec}(M))$	$mn \times mn$
Matrix inner product with G	$G^T\bullet$	$X \mapsto \text{trace}(G^T X)$	$\text{vec}(G)^T$	$1 \times mn$

(We overload $A \otimes B$ to be both the operator and the matrix.)

Table 1: Matrix Operators and the size of their dense representations assuming $X : m \times n, \quad A : n_1 \times n, \quad B : m_1 \times m, \quad M : m \times n, \quad G : m \times n.$

Consider the inner product (matrix dot product)$\langle X, Y \rangle = \text{trace}(X^T Y)$. The identity $\langle X, AY \rangle = \langle A^T X, Y \rangle$ implies $(A_L)^T = (A^T)_L$, in words, the operator adjoint with respect to the left multiplication by A operator is left multiplication by A^T. The operator transposes are $(A_L)^T = (A^T)_L$, $(B_R)^T = (B^T)_R$, and $(M_H)^T = M_H$ (symmetric).

DEFINITION 2. *Let $G^T\bullet$ ("G transpose dot") denote the matrix inner (or dot) product with G. This operator takes a matrix X of the same size as G and returns the scalar,* $G^T \bullet X \equiv Tr(G^T X) = vec(G)^T vec(X) = \sum G_{ij} X_{ij}$.

Why this notation is nice!

LEMMA 4.1. *If the superscript "$()^T$" is overloaded to denote real operator adjoint or matrix transpose as appropriate, $\mathcal{L}$ is a linear operator and G is a matrix, then we have the operator identity: $(\mathcal{L}^T G)^{T\bullet} = G^{T\bullet}\mathcal{L}$. Notice that if we pretend all letters are just matrices and you ignore the dot, the notation has the appearance of the familiar transpose rule.*

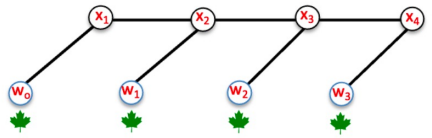
Proof. We have that for all X,

$$(\mathcal{L}^T G)^{T\bullet} X = \langle \mathcal{L}^T G, X\rangle = \langle G, \mathcal{L}X\rangle = G^{T\bullet}\mathcal{L}X,$$

showing that as operators $(\mathcal{L}^T G)^{T\bullet} = G^{T\bullet}\mathcal{L}$. As an example,

$$(A_L^T G)^{T\bullet} = X \mapsto tr((A^T G)^T X), \quad \text{and} \quad G^{T\bullet} A_L = X \mapsto tr(G^T A X),$$

From scalar to vector to matrix networks



Gradient	$\nabla_p \mathcal{L} = M^T \times (I - L)^{-1} \times g$ $= \begin{pmatrix} m_1 & & & \\ & m_2 & & \\ & & m_3 & \\ & & & m_4 \end{pmatrix}^T \times \begin{pmatrix} I & & & \\ -l_2 & I & & \\ & -l_3 & I & \\ & & -l_4 & I \end{pmatrix}^{-T} \times \begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ g_4 \end{pmatrix}$
(i) Scalar $p = w$	$m_i = \delta_i x_{i-1}$ $\quad l_i = \delta_i w_i$ $\quad g_4 = \mathcal{L}'(x_4)$
(ii) Vector $p = [w, b]$	$m_i = [\delta_i x_{i-1} \ \ \delta_i]$ $\quad$ " " $\quad$ " "
(iii) Matrices $p = [W, B]$	$m_i = [\Delta_{iH} \circ X_{i-1R} \ \ \Delta_{iH}]$ $\quad l_i = \Delta_{iH} \circ W_{iL}$ $\quad g_4 = \nabla_{X_4} \mathcal{L}$ ↑ Operators ↑

Table 2: Algebraic Structure for a basic neural network when the parameters are (i) only scalar weights (ii) a weight/bias vector, (iii) a vector of weight/bias matrices. We emphasize the common algebraic structure and the benefit of software that can represent matrices of vectors and matrices of operators.

Neural Net with Operators

$$\begin{pmatrix} dX_1 \\ dX_2 \\ \vdots \\ dX_{N-1} \\ dX_N \end{pmatrix} = \begin{pmatrix} [H_{\Delta_1} \circ R_{X_0}\ H_{\Delta_1}] & & & & \\ & [H_{\Delta_2} \circ R_{X_1}\ H_{\Delta_2}] & & & \\ & & \ddots & & \\ & & & [H_{\Delta_{N-1}} \circ R_{X_{N-2}}\ H_{\Delta_{N-1}}] & \\ & & & & [H_{\Delta_N} \circ R_{X_{N-1}}\ H_{\Delta_N}] \end{pmatrix} \begin{pmatrix} [dW_1; dB_1] \\ [dW_2; dB_2] \\ \vdots \\ [dw_{N-1}; db_{N-1}] \\ [dw_N; db_N] \end{pmatrix}$$

$$+ \begin{pmatrix} 0 & \dots & 0 & 0 & 0 \\ H_{\Delta_2} \circ L_{W_2} & \dots & 0 & 0 & 0 \\ & \ddots & & & \\ & & H_{\Delta_{N-1}} \circ L_{W_{N-1}} & 0 & 0 \\ & & & H_{\Delta_N} \circ L_{W_N} & 0 \end{pmatrix} \begin{pmatrix} dX_1 \\ dX_2 \\ \vdots \\ dX_{N-1} \\ dX_N \end{pmatrix}$$

(Generally triangular!)

```
## Operators
ℒ(A::Matrix)  = Operator(X->A*X)
ℛ(A::Matrix)  = Operator(X->X*A)
ℋ(A::Matrix)  = Operator(X->X.*A)
ℐ()   =          Operator(X->X)
𝒪()   =          Operator(X->zero(X))
```


```
∇J = D' * (ImL'\g)
```

Demo

Some CS points

Modern Computer Science Meets Linear Algebra:

- Generic Programming
  - Note: backslash is not lapack's backslash which isn't generic
  - Composablity
- Transpose all the way down
  - Contraversy: `https://discourse.julialang.org/t/why-is-transpose-recursive/2550`
- Performance

Abstraction

… mathematics is the art of giving the same name to different things. It is enough that these things, though differing in matter, should be similar in form, to permit of their being, so to speak, run in the same mould. When language has been well chosen, one is astonished to find that all demonstrations made for a known object apply immediately to many new objects: nothing requires to be changed, not even the terms, since the names have become the same.

Henri Poincaré
over 100 years ago

Conclusion:

BACKPROPAGATION THROUGH BACK SUBSTITUTION WITH A BACKSLASH

ALAN EDELMAN*, EKIN AKYÜREK†, AND YUYANG WANG‡