d²p: Structured Soft Attention Is All You Need

*Differentiable dynamic programming **is** structured attention, and you can learn its hyperparameters end-to-end.*

Casey Sumagaysay Mogilevsky · Kimberly Liang

ICML 2026

Same recurrence relation, a tale of two operators: max vs log-sum-exp

setup

max

log-sum-exp (T)

hard alignment

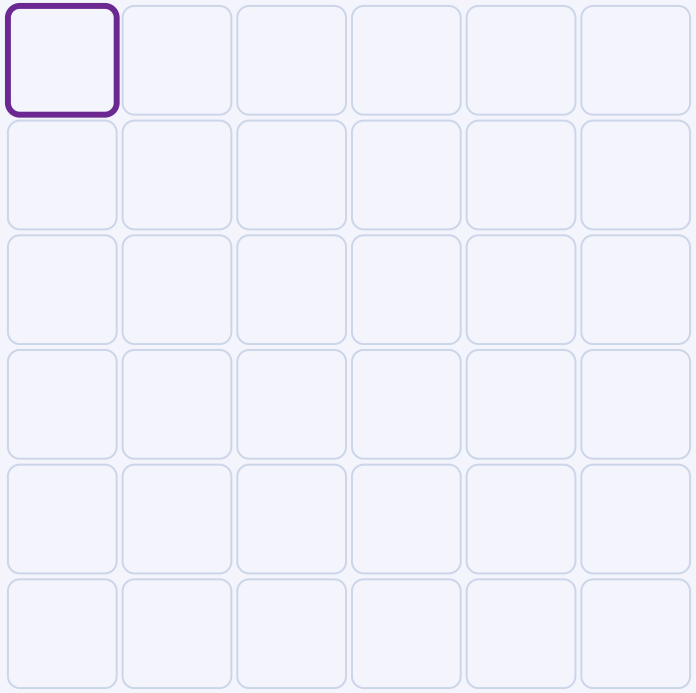
differentiable alignment

Same score matrix S. Same antidiagonal schedule. Only one operator differs.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

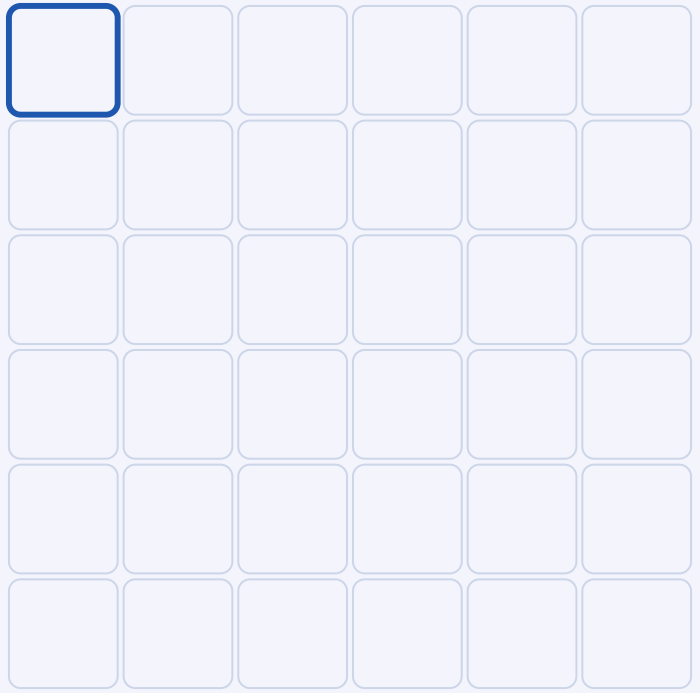
forward sweep · antidiagonal **1** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



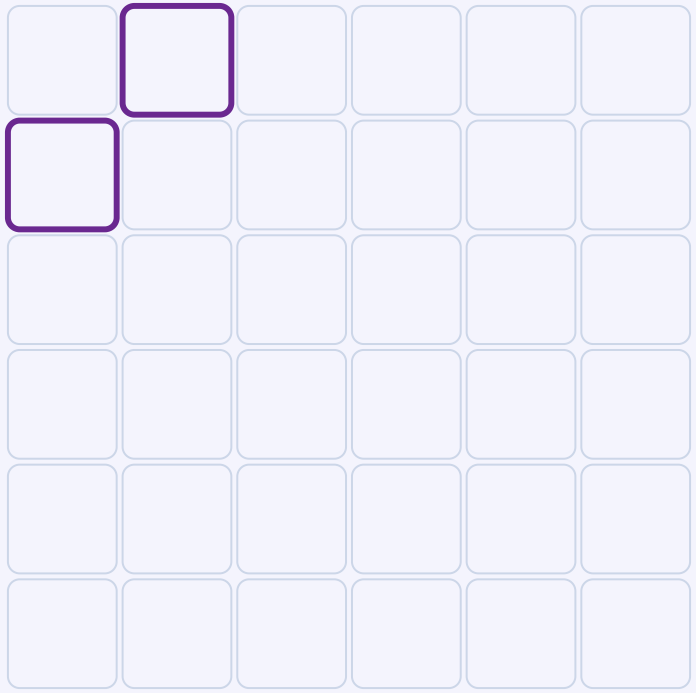
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

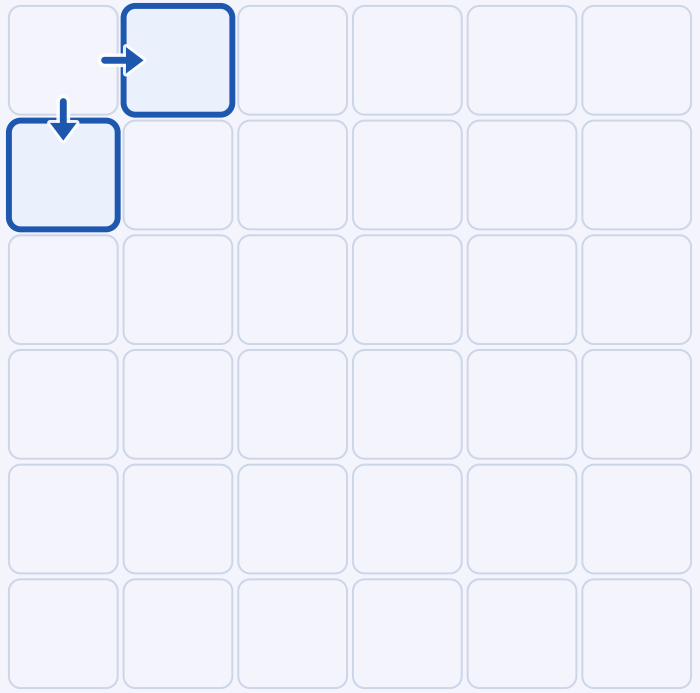
forward sweep · antidiagonal **2** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



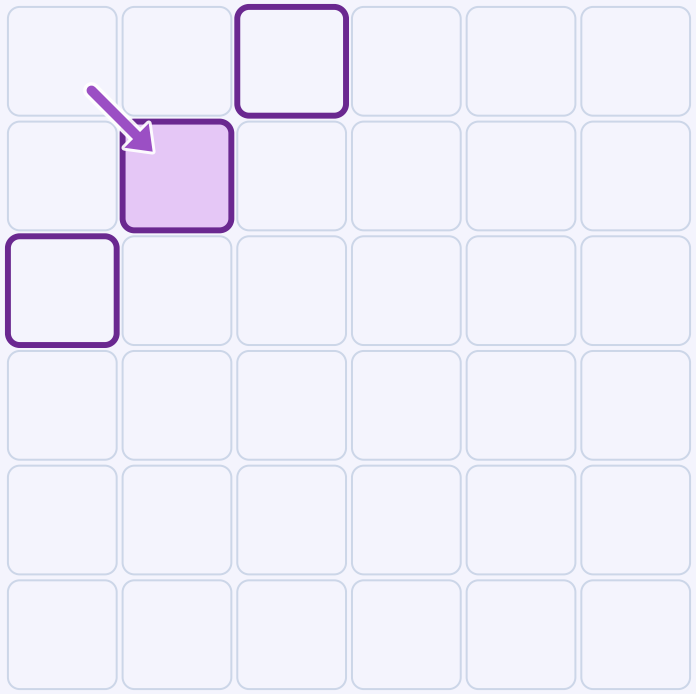
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

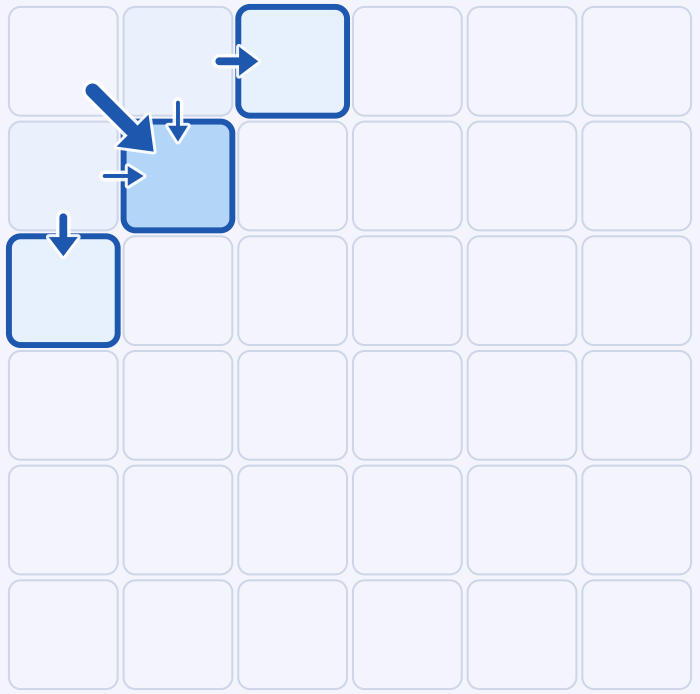
forward sweep · antidiagonal **3** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



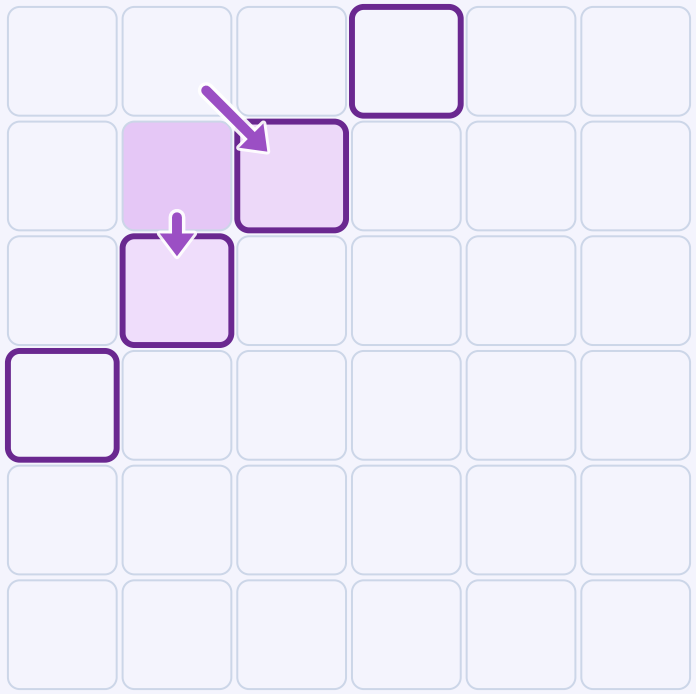
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

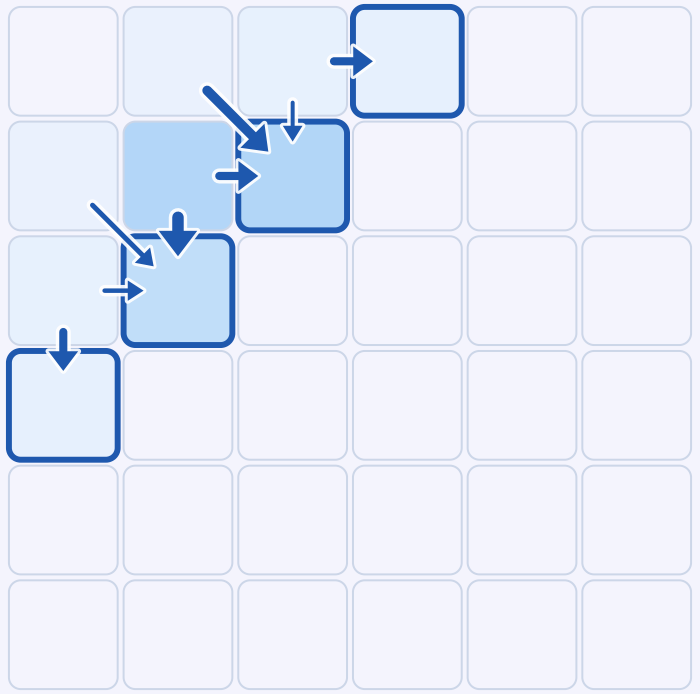
forward sweep · antidiagonal **4** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



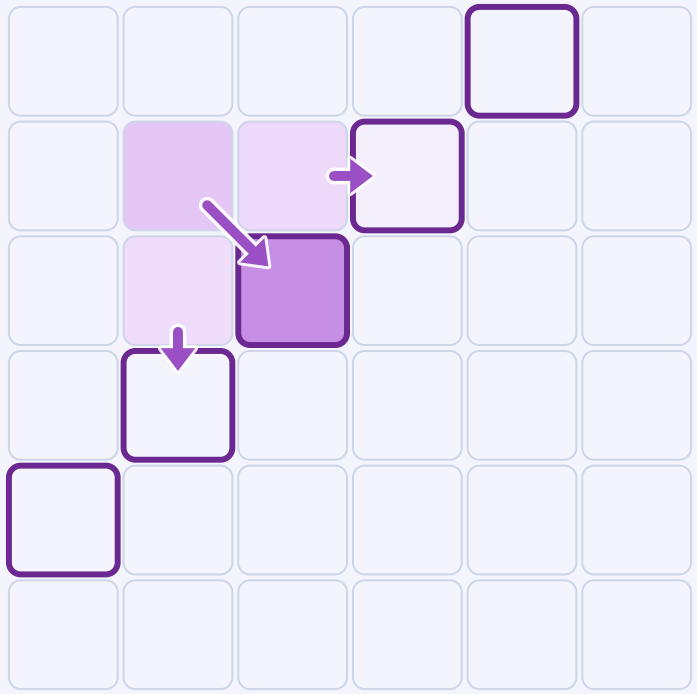
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

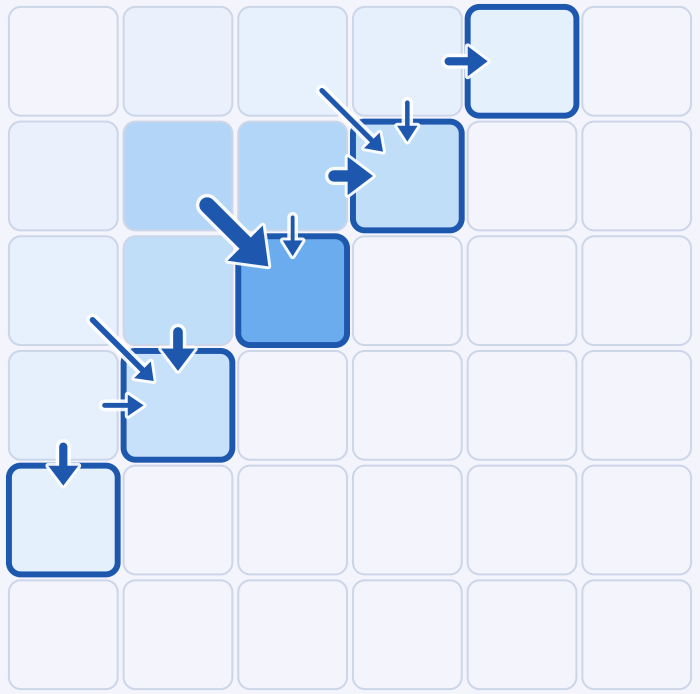
forward sweep · antidiagonal **5** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



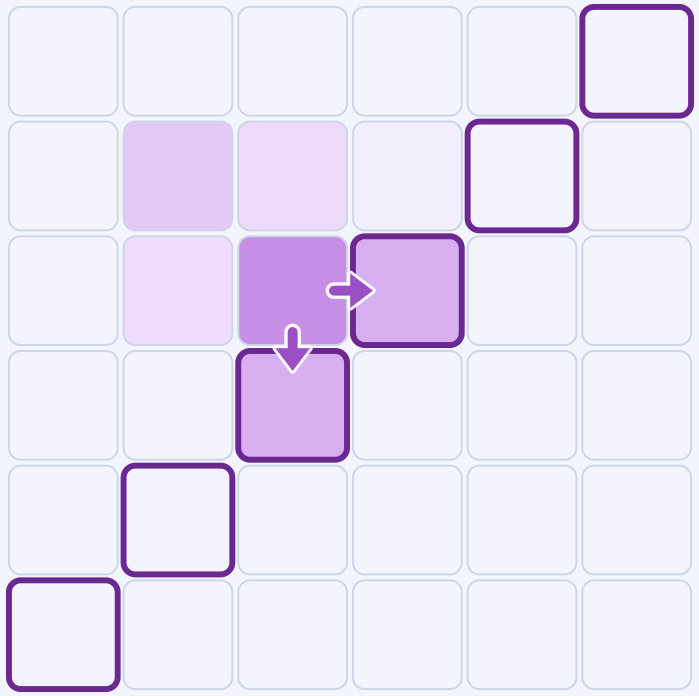
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

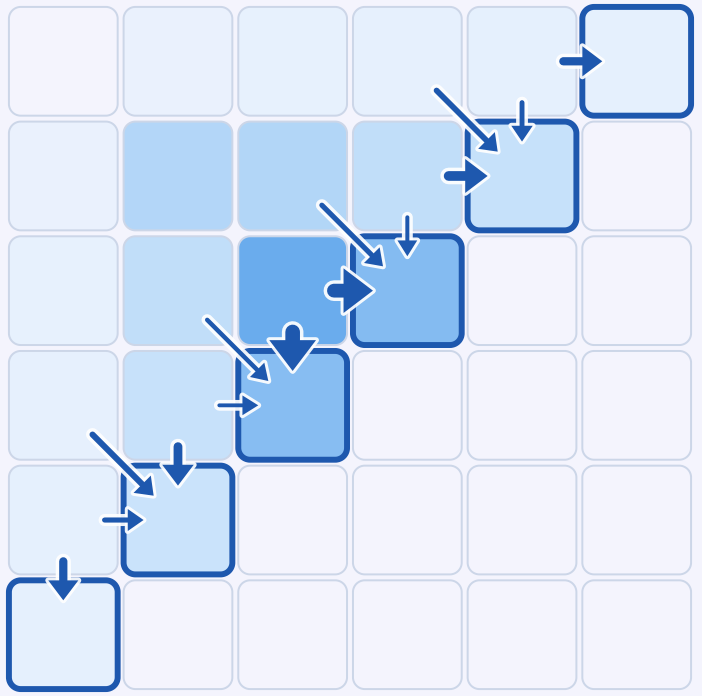
forward sweep · antidiagonal **6** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



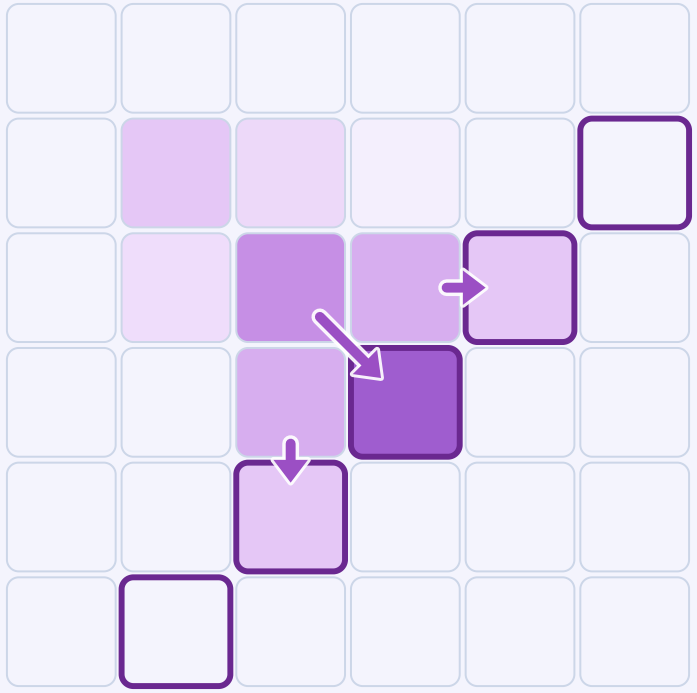
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

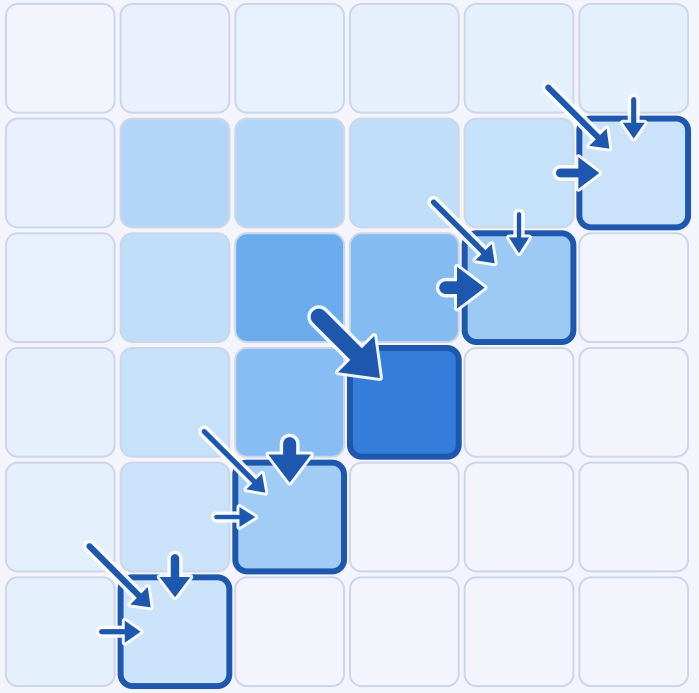
forward sweep · antidiagonal **7** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



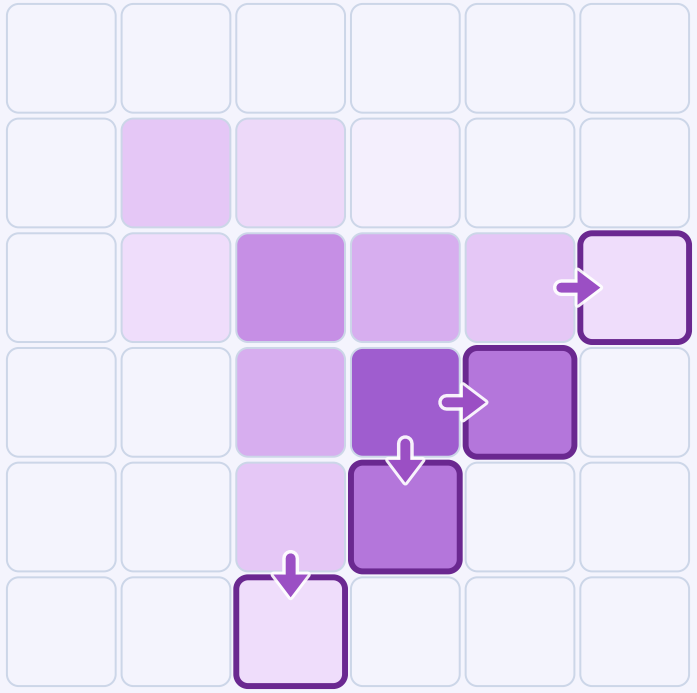
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

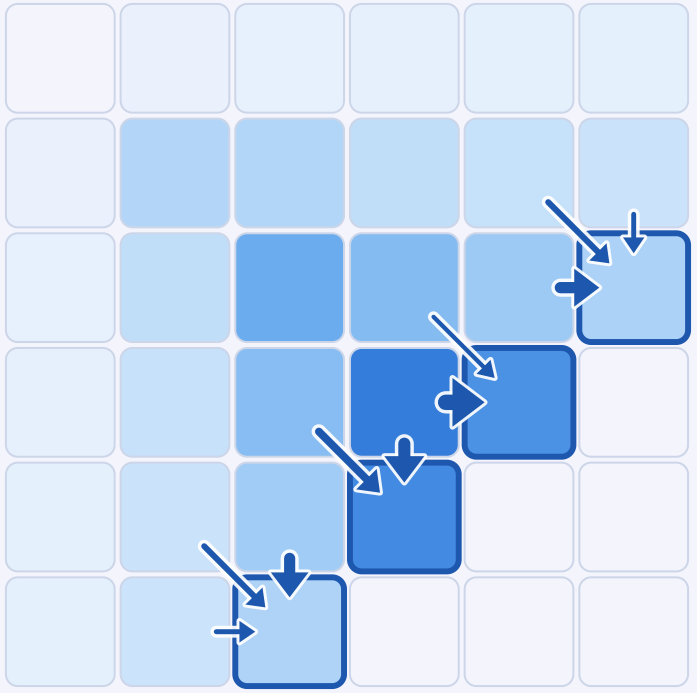
forward sweep · antidiagonal **8** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



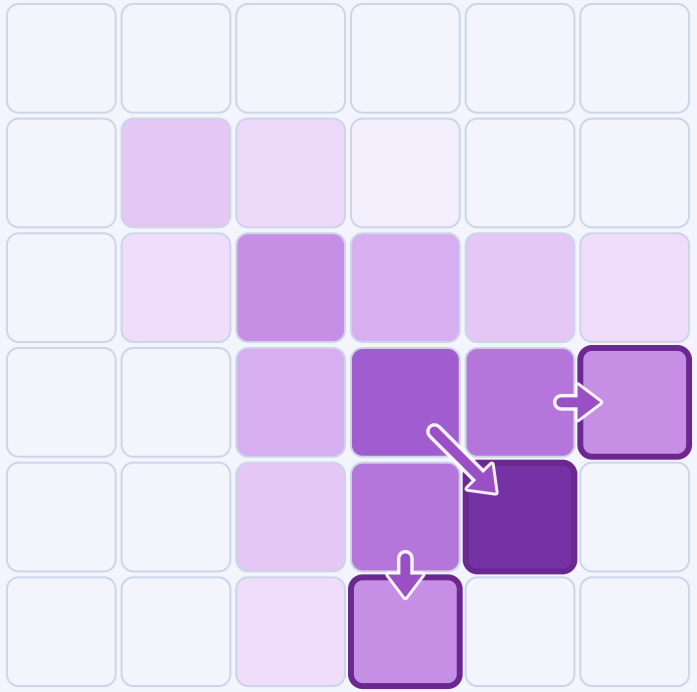
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

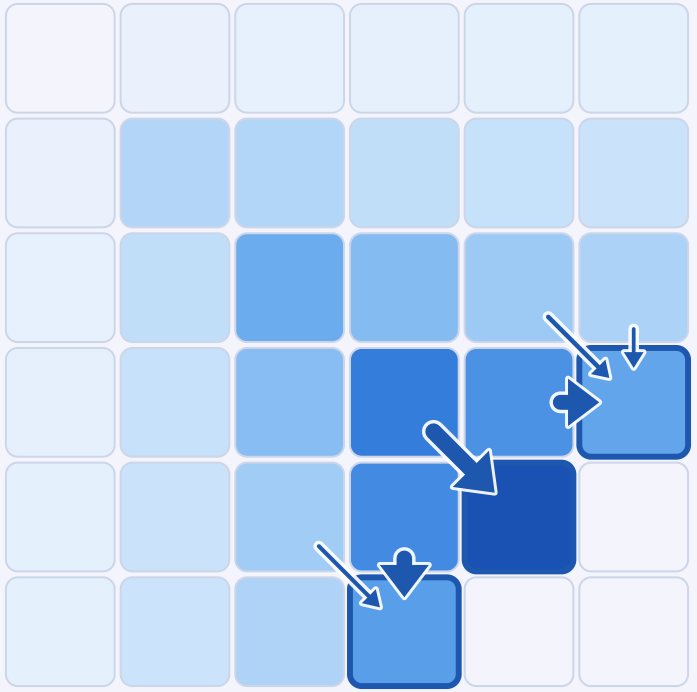
forward sweep · antidiagonal **9** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



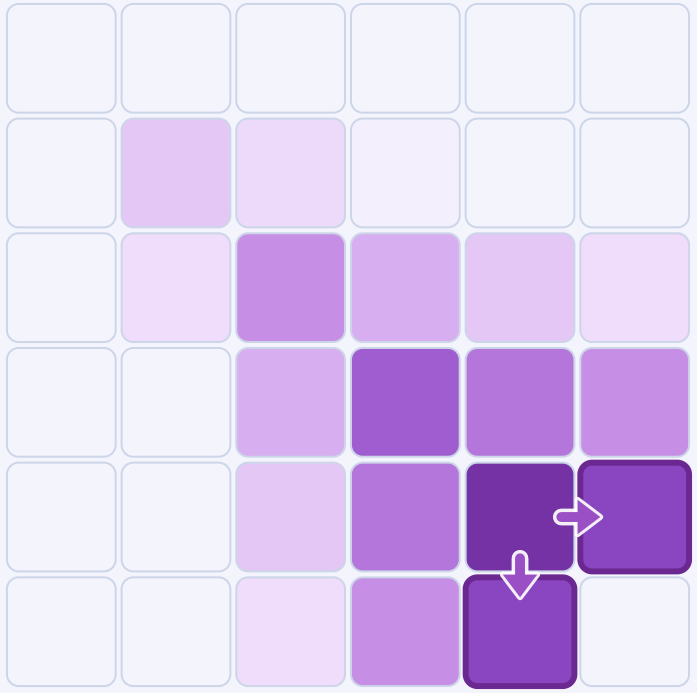
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

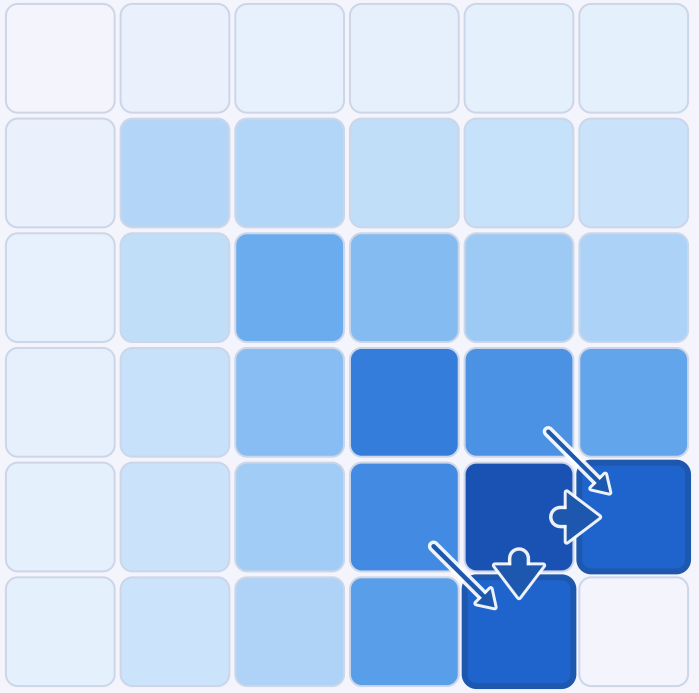
forward sweep · antidiagonal **10** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



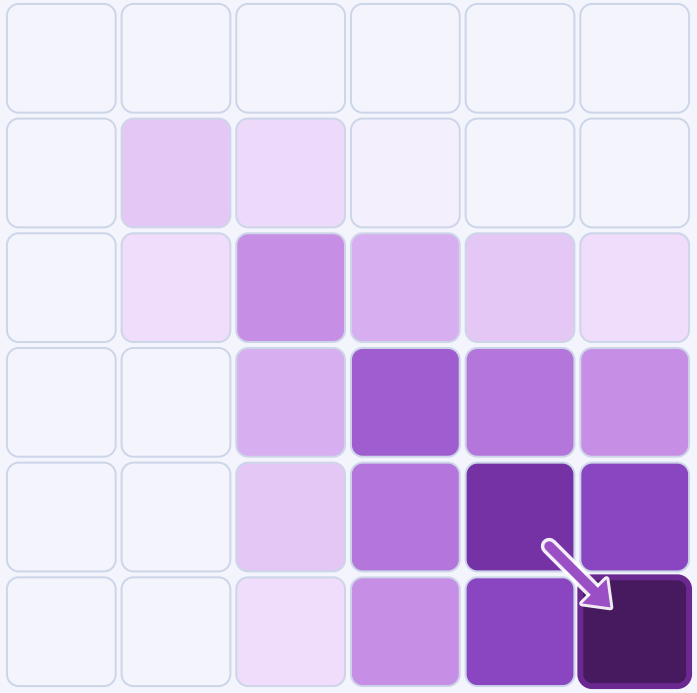
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

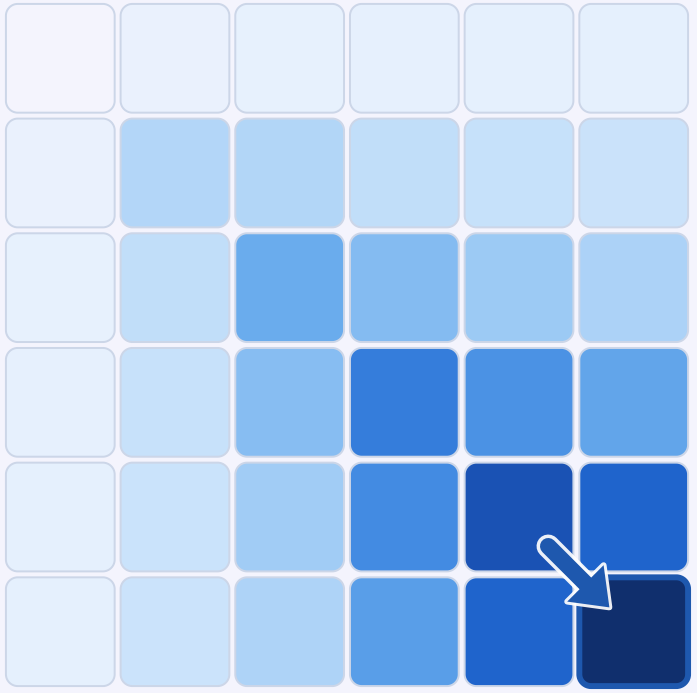
forward sweep · antidiagonal **11** of 11

max



max commits to one path: one argmax pointer per cell

log-sum-exp (T)



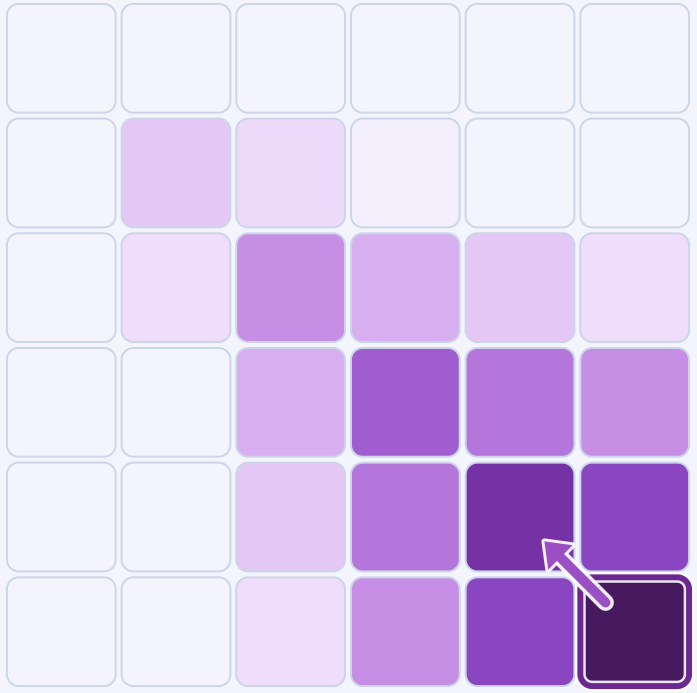
log-sum-exp blends all possible paths: weighted over all predecessors

The wavefront schedule (k = i + j) is shared, which is why one kernel parallelizes both.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

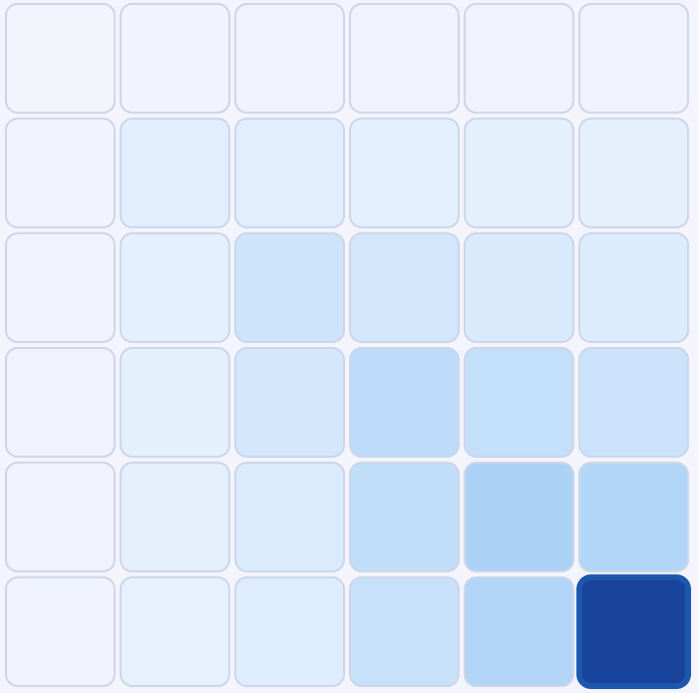
backward sweep · antidiagonal **11** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



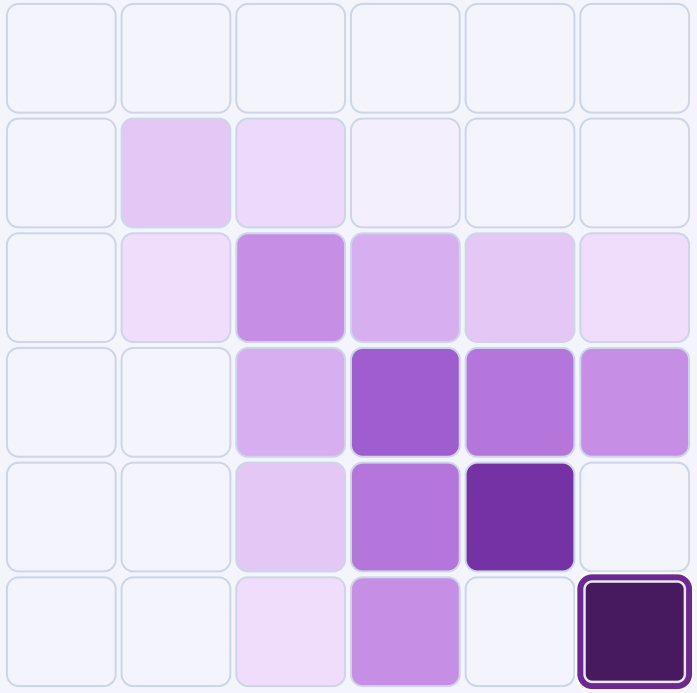
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

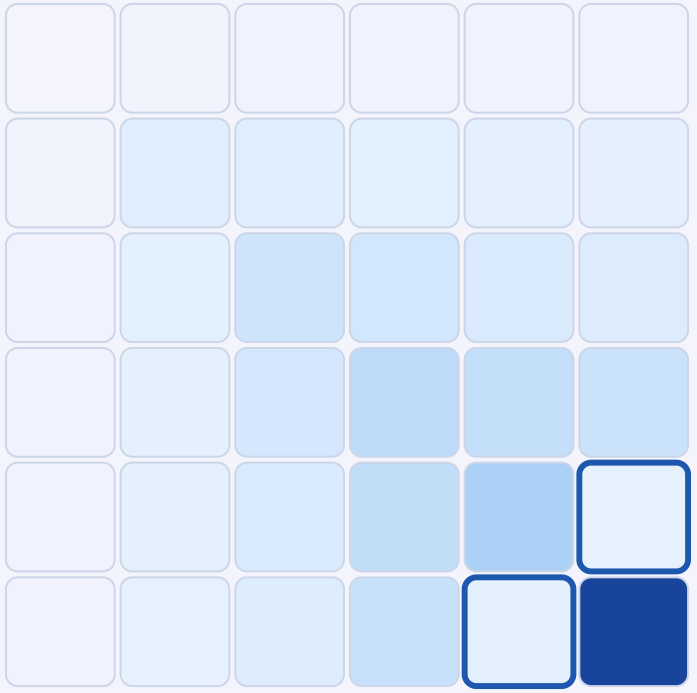
backward sweep · antidiagonal **10** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



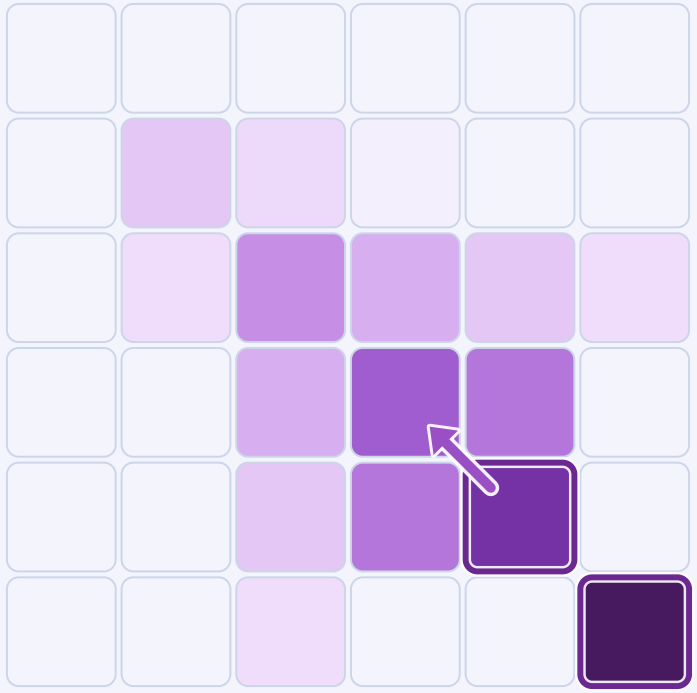
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

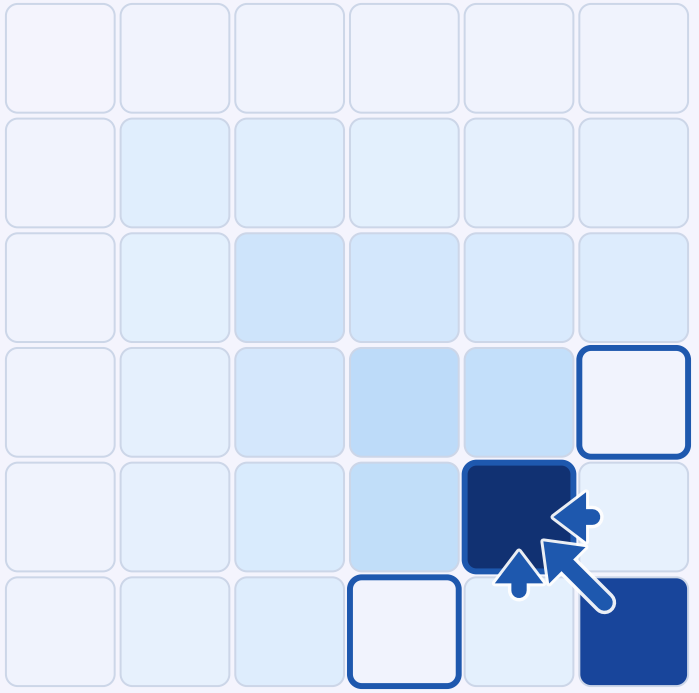
backward sweep · antidiagonal **9** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



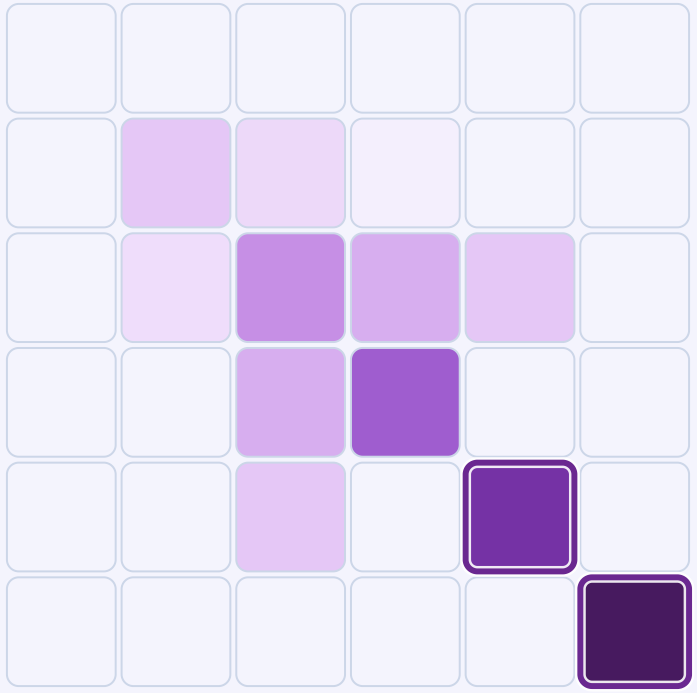
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

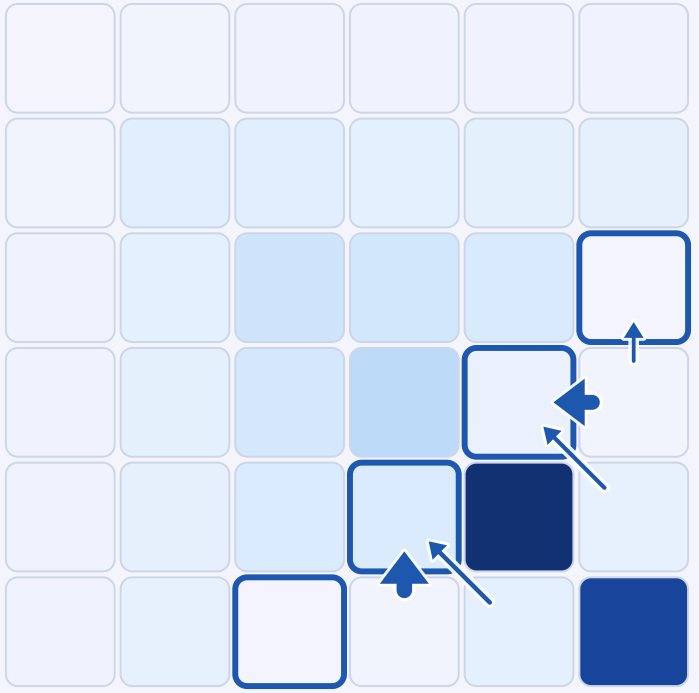
backward sweep · antidiagonal **8** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



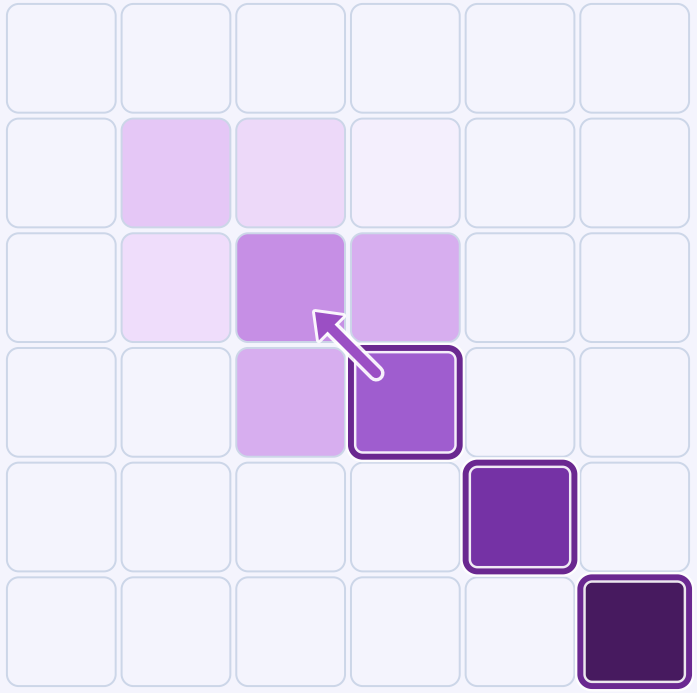
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

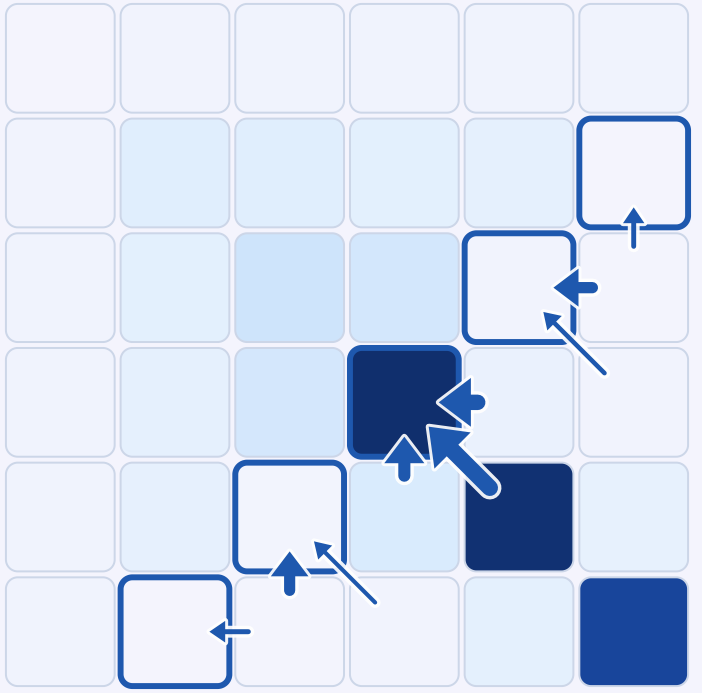
backward sweep · antidiagonal **7** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



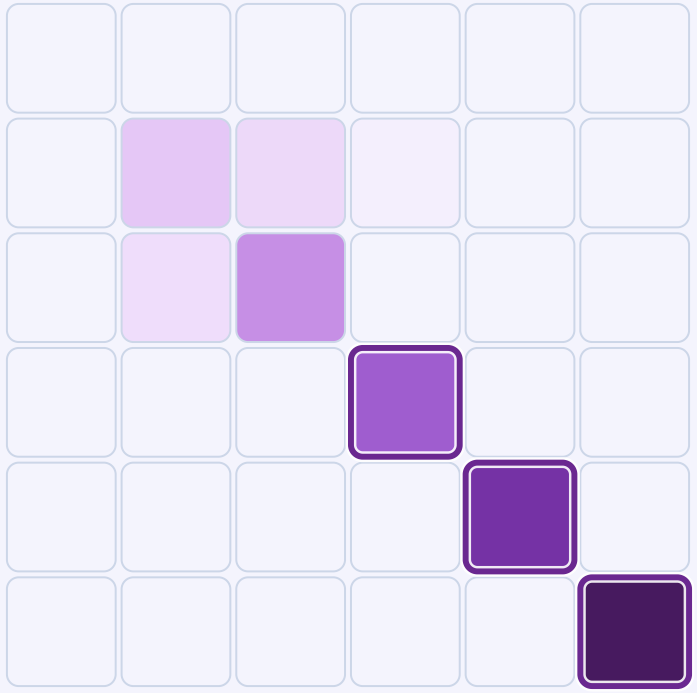
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

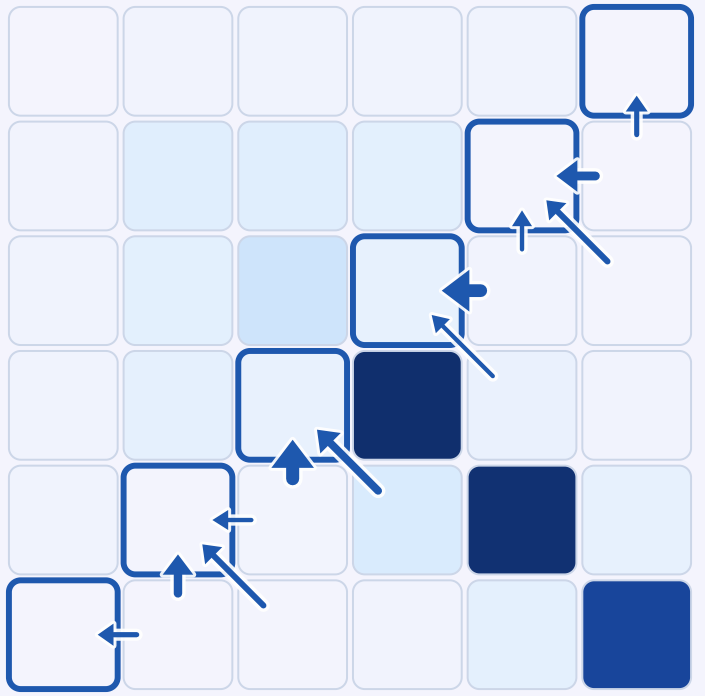
backward sweep · antidiagonal **6** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



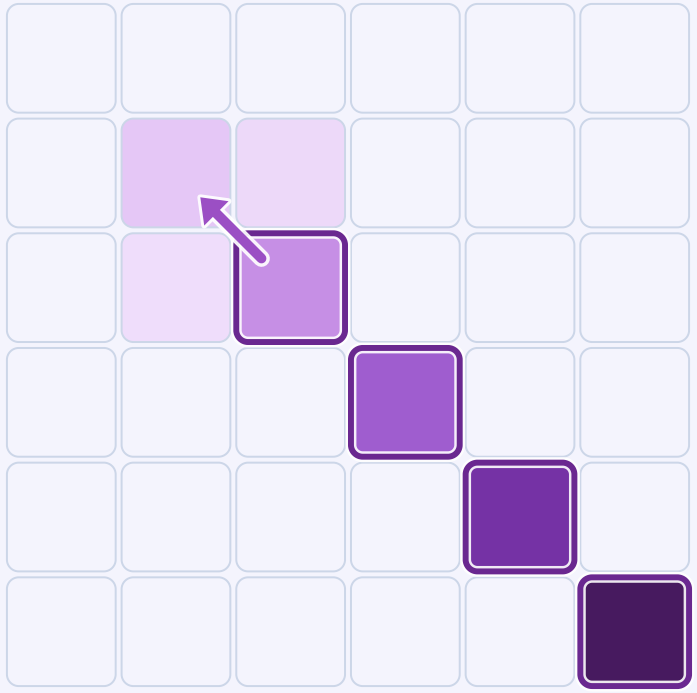
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

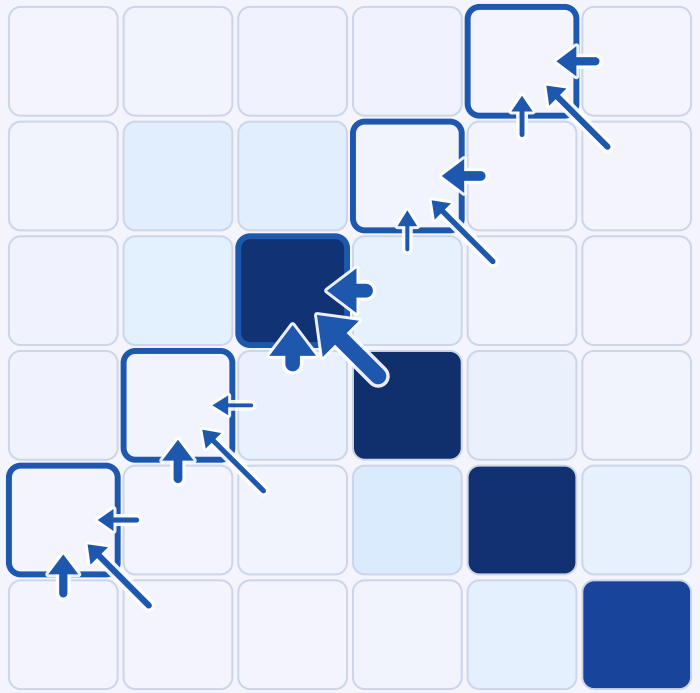
backward sweep · antidiagonal **5** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



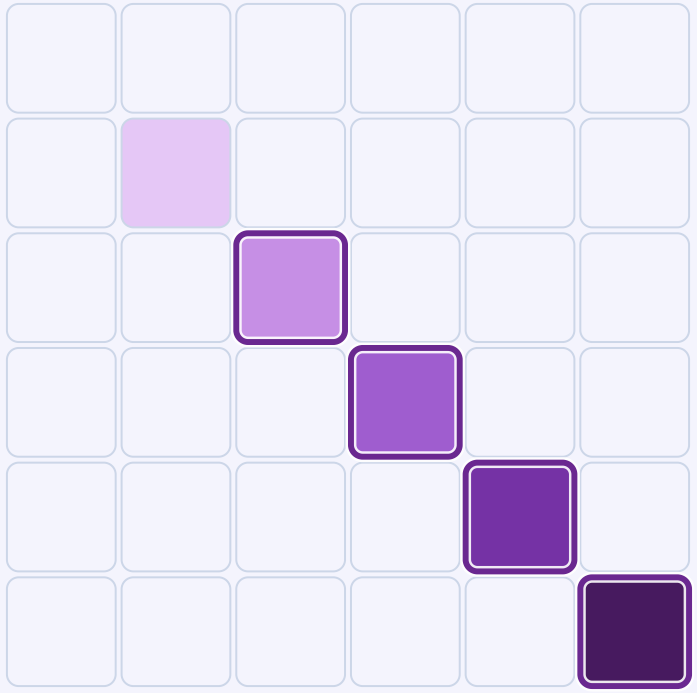
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

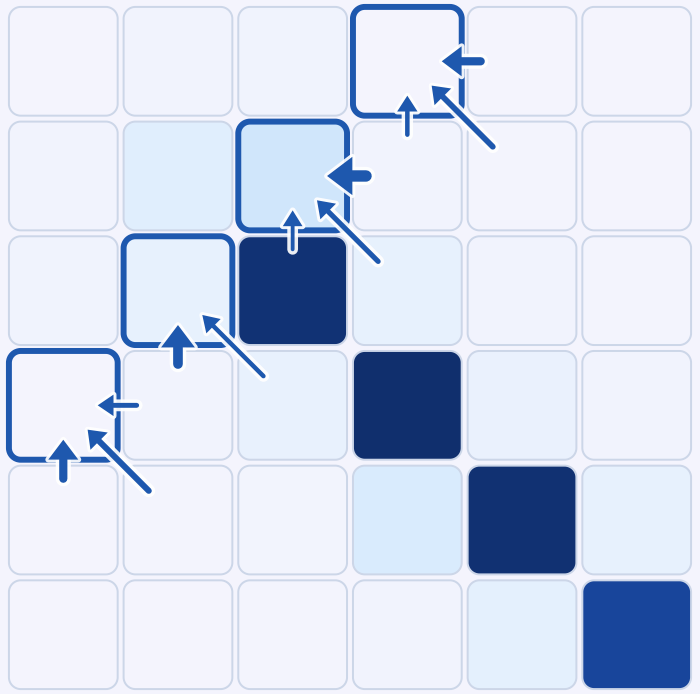
backward sweep · antidiagonal **4** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



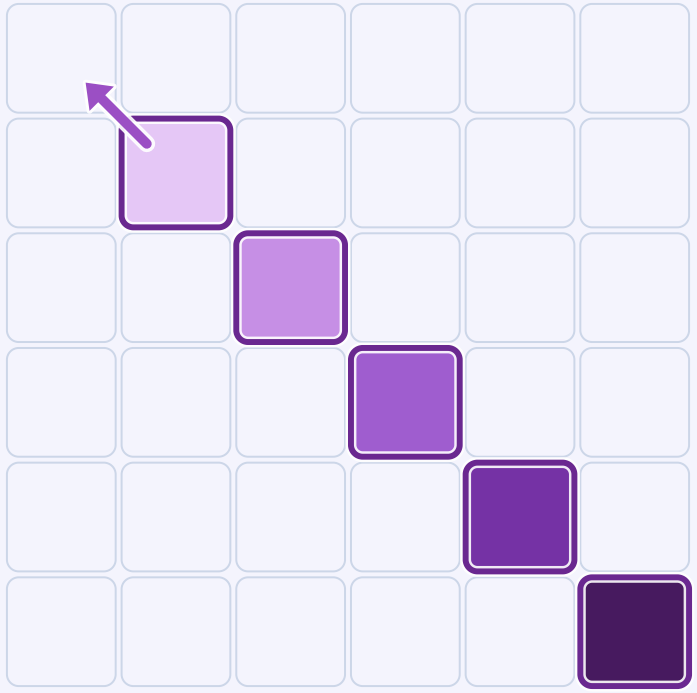
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

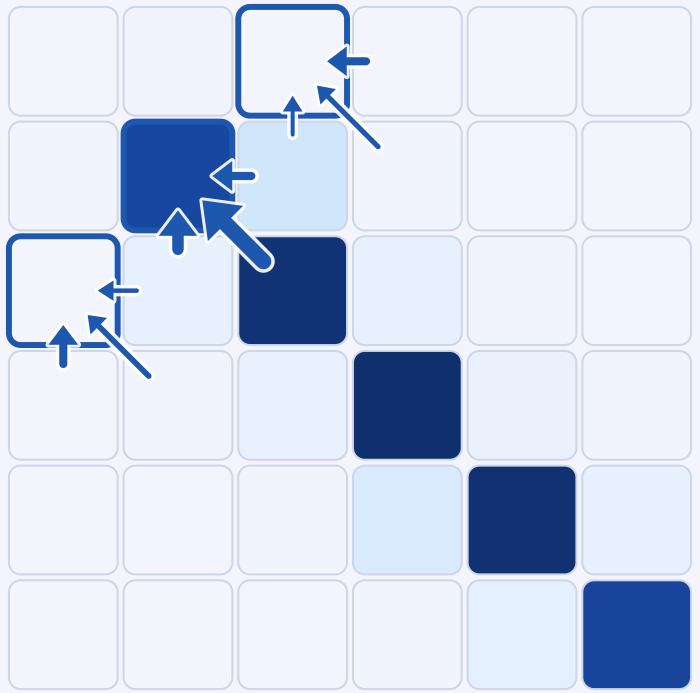
backward sweep · antidiagonal **3** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



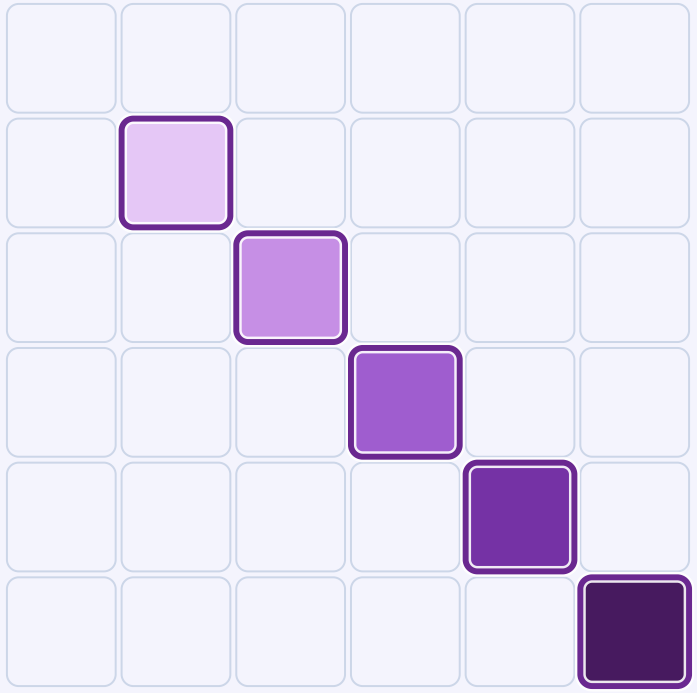
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

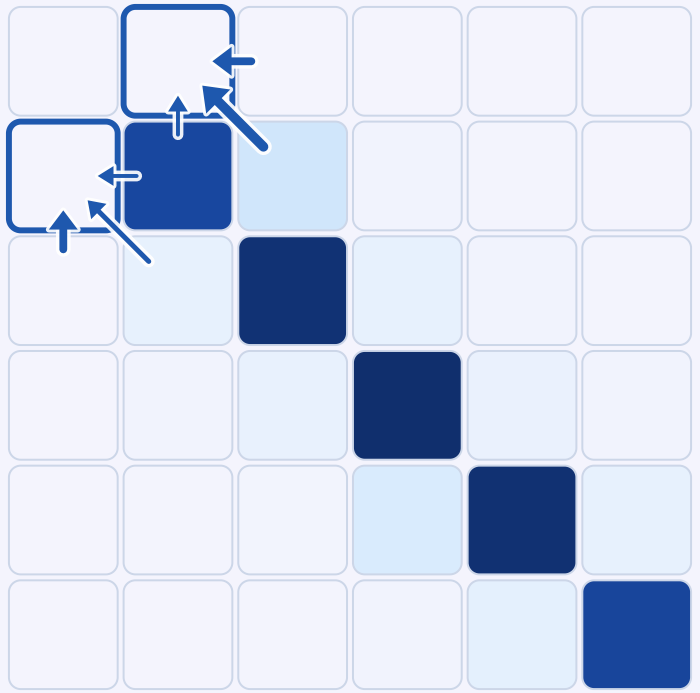
backward sweep · antidiagonal **2** of 11

max



traceback: follow the one pointer back

log-sum-exp (T)



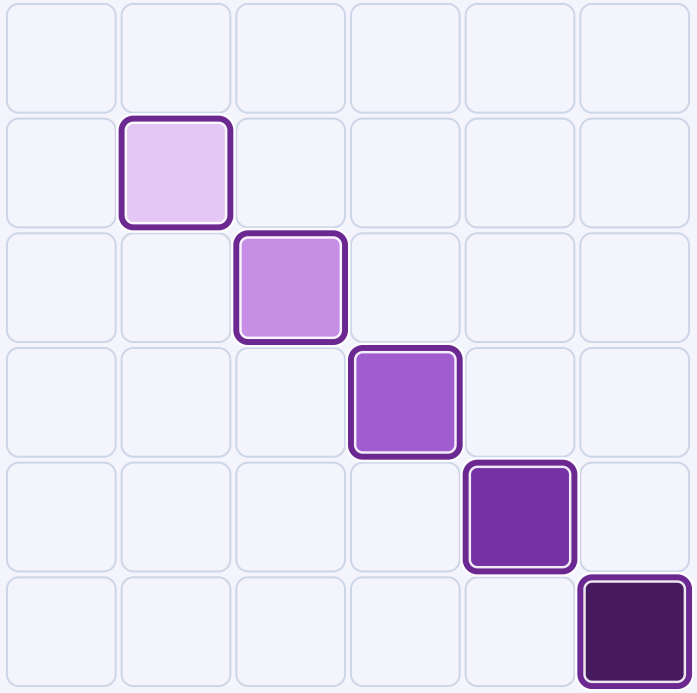
gradient: credit flows back, weighted (the marginals)

The same schedule in reverse: max's gradient is one path, log-sum-exp's gradient is the whole attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

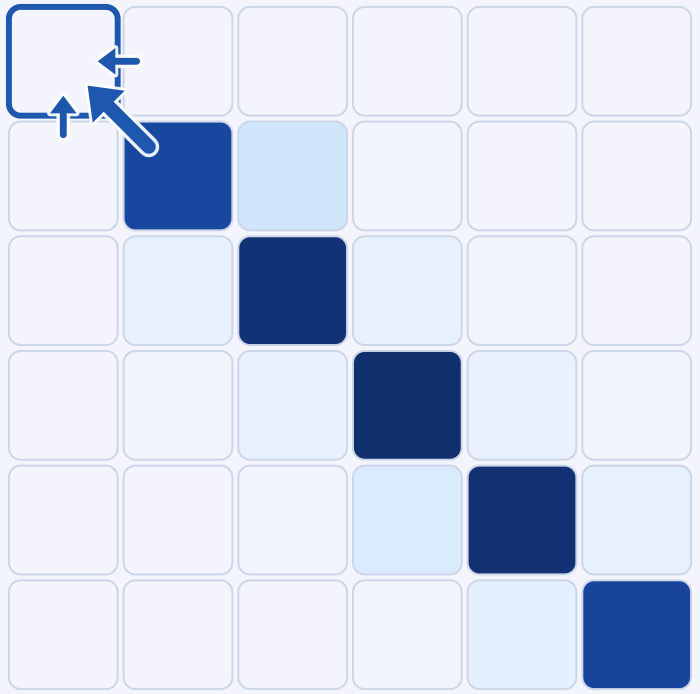
readout

max



traceback → one alignment (= the gradient of the max)

log-sum-exp (T)



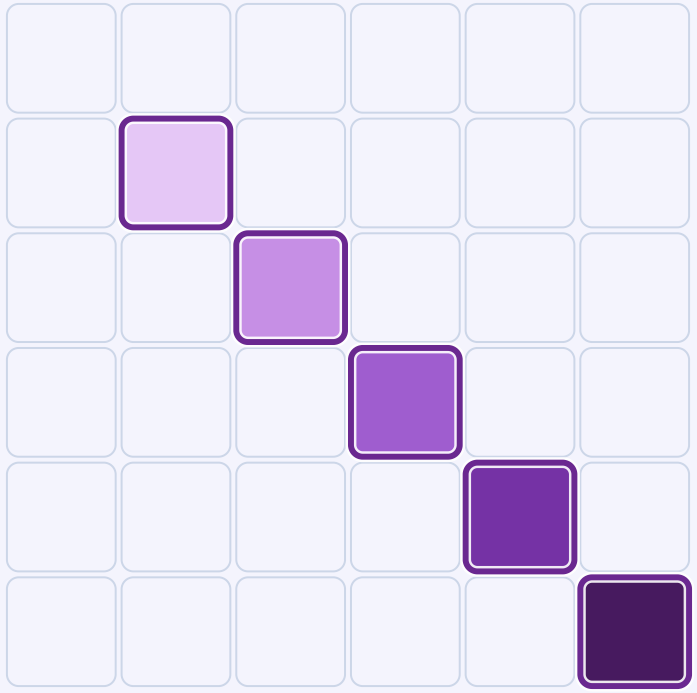
marginals → an attention map (= $\partial V_T / \partial S$)

Soften the operator and the discrete traceback becomes a differentiable attention map.

Same recurrence relation, a tale of two operators: max vs log-sum-exp

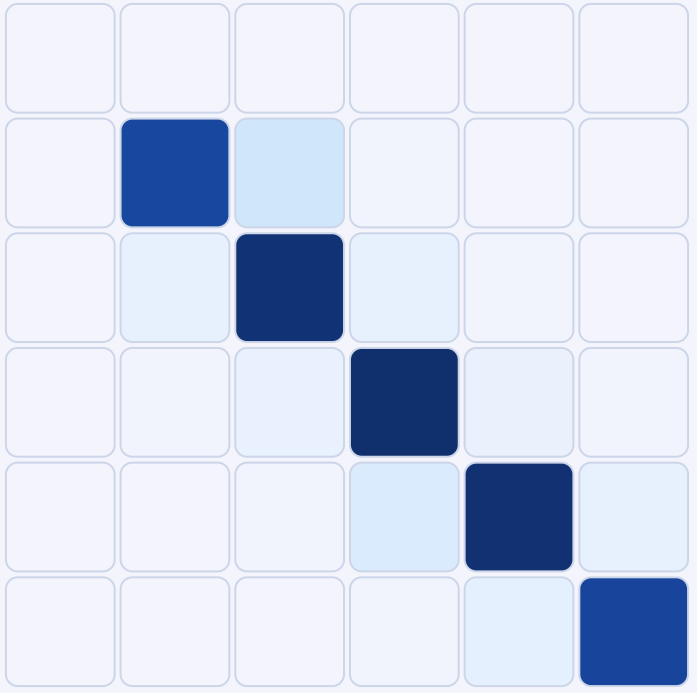
outputs

max



one alignment path

log-sum-exp (T)



one attention map (all paths, weighted)

max returns a single alignment; log-sum-exp returns the whole differentiable attention map.

Differentiable dynamic programming is structured attention at finite temperature

T = 16

Softmax

SW affine

CKY

high temperature: diffuse, high-entropy attention

Differentiable dynamic programming is structured attention at finite temperature

T = 4

Softmax

SW affine

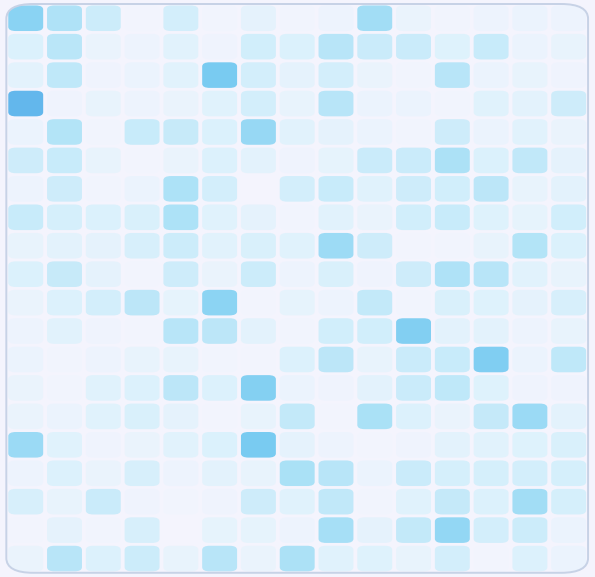
CKY

sharpening as the temperature falls

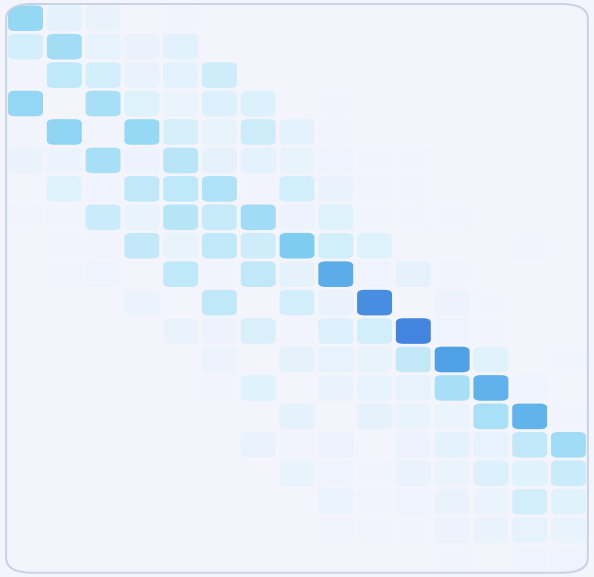
Differentiable dynamic programming is structured attention at finite temperature

T = 1

Softmax



SW affine



CKY

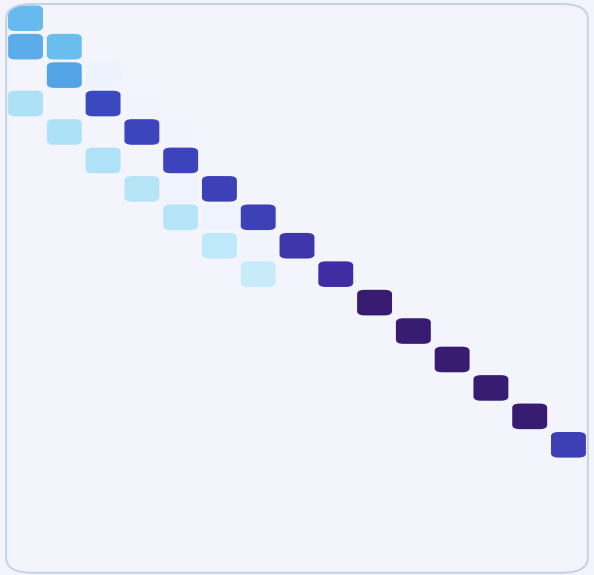
sharpening as the temperature falls

Differentiable dynamic programming is structured attention at finite temperature

T = 0.25

Softmax

SW affine



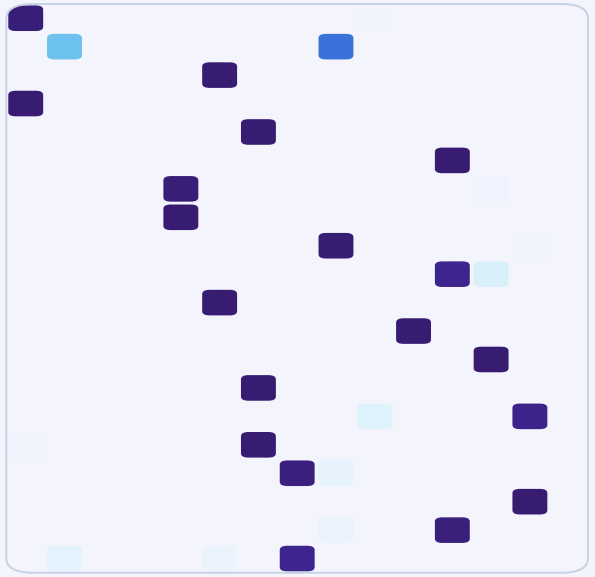
CKY

sharpening as the temperature falls

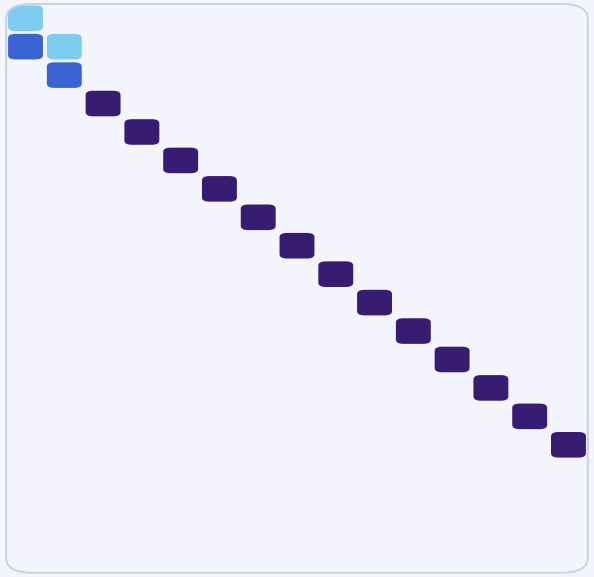
Differentiable dynamic programming is structured attention at finite temperature

T = 0.0625

Softmax



SW affine



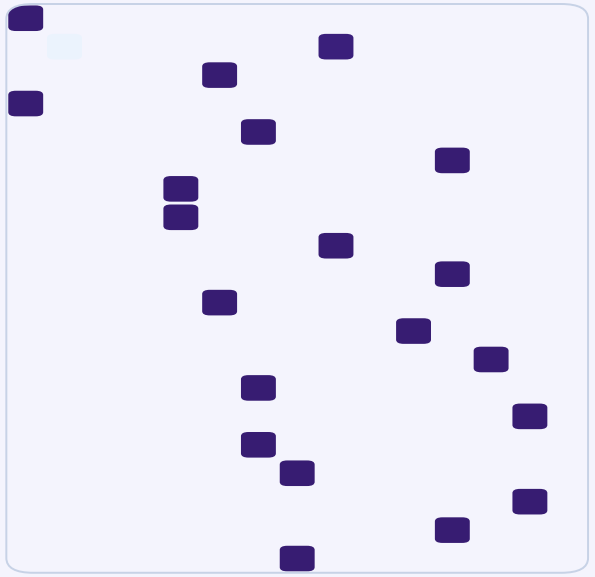
CKY

low temperature: nearly the classical algorithm

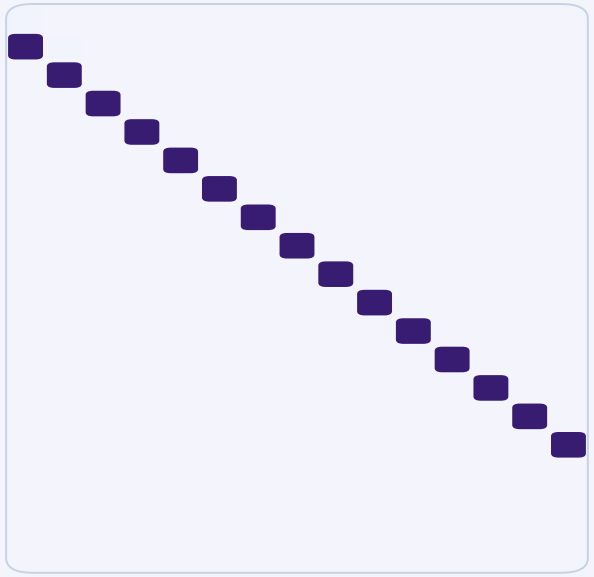
A dynamic program is structured hard attention at T = 0

T = 0 (argmax)

Softmax



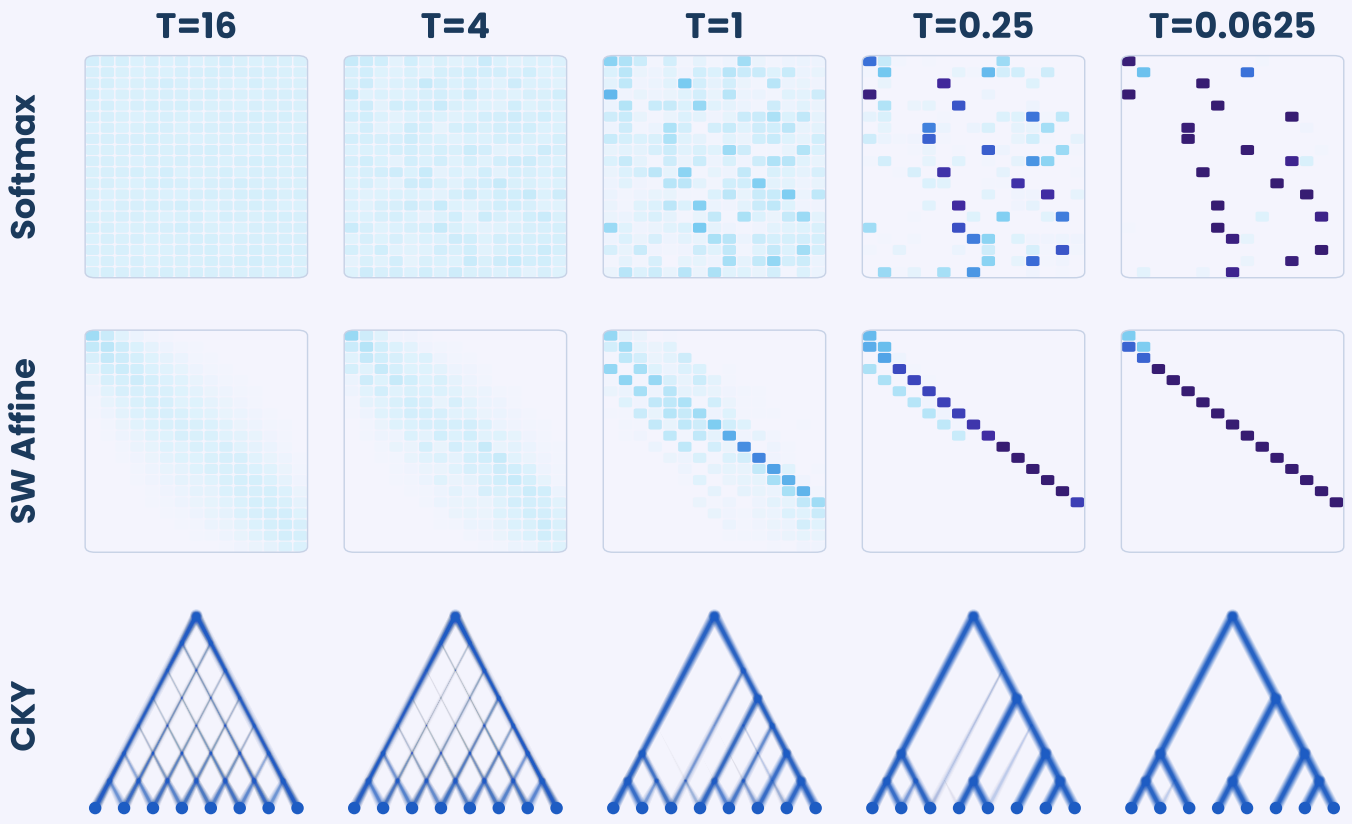
SW affine



CKY

the operator becomes argmax: attention collapses to the one-hot classical algorithm

Differentiable dynamic programming is structured attention at finite temperature



The marginal *is* the attention map *is* the gradient, diffuse at high entropy, the classical algorithm at low.

Learning through the algorithm itself is a second-order problem

The marginals *are* the first derivative $P_T = \partial V_T / \partial S$.
Going one derivative further differentiates them.

order 1: $\partial V_T / \partial S = P_T$ marginals = attention (known)

order 2: $\partial P_T / \partial S$ (encoder) & $\partial P_T / \partial \boldsymbol{\eta}$ (parameters) **this work**

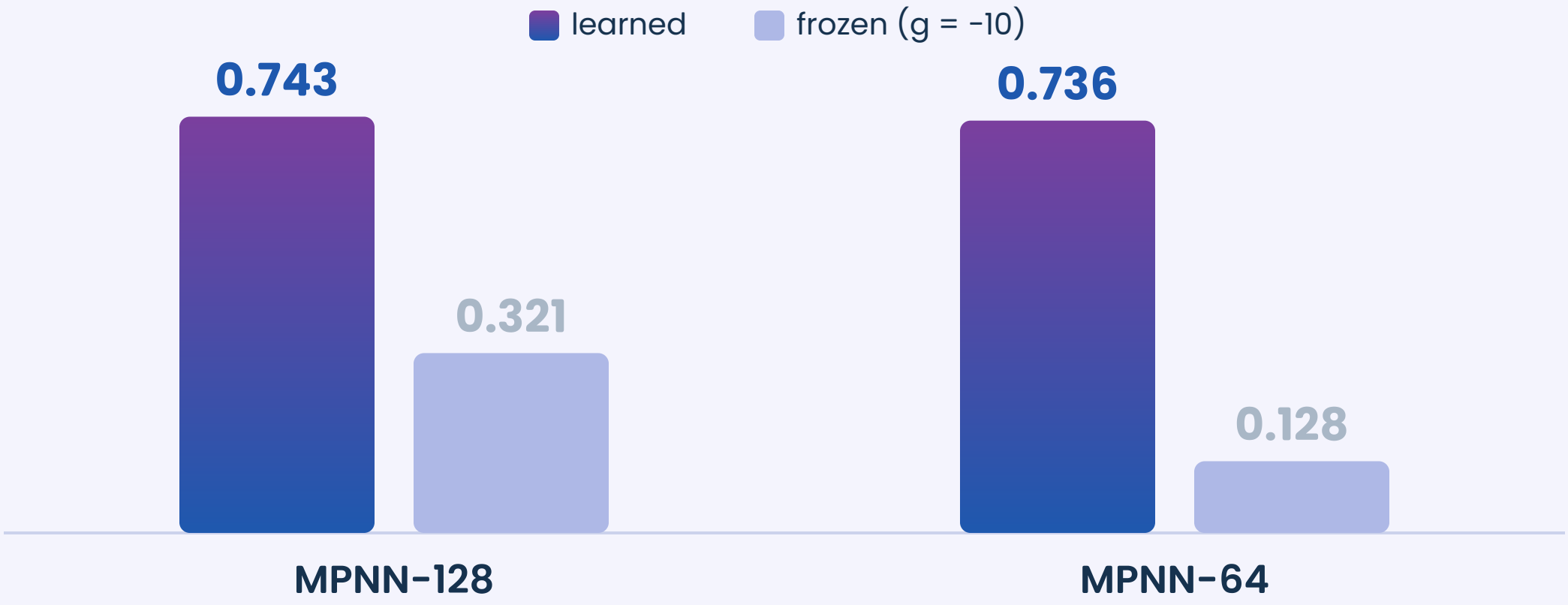
$$\frac{\partial P_{T,ij}}{\partial \boldsymbol{\eta}} = \frac{1}{T} \operatorname{Cov}_{p_T}\big(\mathbf{1}\{(i,j) \in Y\},\, \varphi_{\boldsymbol{\eta}}(Y)\big)$$

a closed-form Gibbs covariance, for all twelve algorithms

the gradient that tunes the ubiquitous $g = -10$.

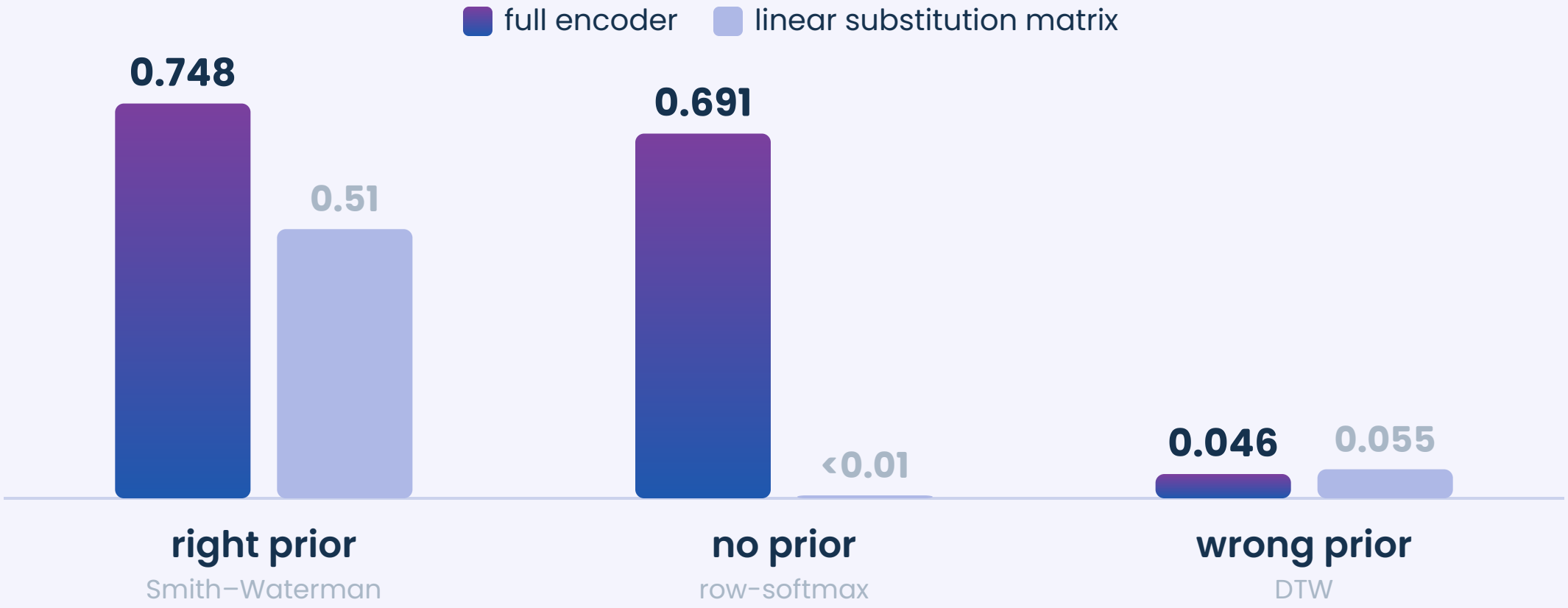
Second-order terms are not optional

Freeze the linear Smith–Waterman gap at the default $g = -10$ and F_1 collapses — worst where the encoder is smallest; learning it through the cross-Jacobian recovers full performance.



It is the recursion, not the capacity, that decides

Same encoder, same training. Swap only the prior.



Capacity can substitute for absent structure, but cannot repair the wrong one.

One mechanism, opposite ends of the structure–data spectrum

Protein structure alignment

data-scarce

91%

of the TM-align lDDT ceiling

($F_1 = 0.748$, lDDT 0.445)

0.748 vs **0.468**

beating a re-learned Foldseek 3Di

One mechanism, opposite ends of the structure–data spectrum

Protein structure alignment

data-scarce

91%

of the TM-align lDDT ceiling

($F_1 = 0.748$, lDDT 0.445)

0.748 vs **0.468**

beating a re-learned Foldseek 3Di

Constituency parsing

data-abundant

0.003

F_1 from dense per-span supervision

(English; a structured CKY CRF)

no structural supervision

the same machinery, unchanged

Sutton's bitter lesson, sharpened

Hand-tuned *structure* loses to scale, worst where data can learn it.

d²p fixes only the structure inside attention, and learns every parameter in it.

So its one commitment is the one capacity cannot recover.

Sutton's Bitter Lesson is least bitter where structure is rich and data scarce.

Chief among these domains: the natural sciences.