Preconditioned DeltaNet: A Curvature and Online-Regression View

Paper. These notes follow Tumma, Loo, and Rus, *Preconditioned DeltaNet: Curvature-aware Sequence Modeling for Linear Recurrences*, arXiv:2604.21100, especially Sections 2.3 and 3.1–3.5. The aim is to explain the paper from the following point of view:

Delta rule = online first-order regression; preconditioning = curvature-aware online regression.

1 Test-time regression: the state as a learned map

The paper's starting point (Section 2.3, Eq. (1)) is to interpret the recurrent state

$$S \in \mathbb{R}^{d_v \times d_k}$$

as a linear map from key space to value space,

$$S : \mathbb{R}^{d_k} \longrightarrow \mathbb{R}^{d_v}, \qquad k \mapsto Sk.$$

Given key–value pairs (k_i, v_i) seen up to time t, the ideal state is the ridge-regression solution

$$L_t(S) = \frac{1}{2}\sum_{i=1}^{t} \|Sk_i - v_i\|_2^2 + \frac{\lambda}{2}\|S\|_F^2, \qquad S_t^* \in \arg\min_S L_t(S). \tag{P1 / paper Eq. 1}$$

For clarity, first ignore the ridge term. For one summand

$$\ell_i(S) = \frac{1}{2}\|Sk_i - v_i\|^2,$$

a perturbation $S \mapsto S + \Delta S$ gives

$$\ell_i(S + \Delta S) = \ell_i(S) + (Sk_i - v_i)^\top \Delta S\, k_i + O(\|\Delta S\|^2).$$

Hence

$$\nabla_S \ell_i(S) = (Sk_i - v_i)k_i^\top,$$

and therefore

$$\begin{aligned} \nabla_S L_t(S) &= \sum_{i=1}^{t} (Sk_i - v_i)k_i^\top \\ &= S\underbrace{\left(\sum_{i=1}^{t} k_i k_i^\top\right)}_{G_t} - \underbrace{\left(\sum_{i=1}^{t} v_i k_i^\top\right)}_{C_t}. \end{aligned}$$

Including ridge regularization replaces G_t by

$$G_t = \sum_{i=1}^{t} k_i k_i^\top + \lambda I = K_t^\top K_t + \lambda I, \qquad C_t = \sum_{i=1}^{t} v_i k_i^\top = V_t^\top K_t.$$

Thus

$$\boxed{\nabla L_t(S) = SG_t - C_t.}$$

Setting the gradient to zero gives the normal equations (paper Eq. (2))

$$\boxed{S_t^* G_t = C_t, \qquad S_t^* = C_t G_t^{-1}.} \tag{P2 / paper Eq. 2}$$

The role of the ridge term is precisely to make this solution well-posed: with $\lambda > 0$ the Gram matrix $G_t = K_t^\top K_t + \lambda I$ is strictly positive definite (hence invertible) for every t, so S_t^* exists and is unique even early in the sequence when $t < d_k$ and the keys have not yet spanned key space—exactly the regime where the unregularized $\sum_i k_i k_i^\top$ is singular. Equivalently, the $\frac{\lambda}{2}\|S\|_F^2$ penalty discourages large solutions: writing the SVD $K_t = U\Sigma V^\top$ with singular values σ_j, ridge shrinks each key-direction component of S_t^* by the factor $\sigma_j^2/(\sigma_j^2 + \lambda) < 1$ relative to the unregularized least-squares solve. The shrinkage is hardest in the under-observed directions (small σ_j)—precisely those in which the unregularized solve would otherwise blow up, since it caps the gain $1/\sigma_j^2$ at $1/\lambda$. Invertibility and bounded solution norm are thus two readings of the same floor λ that $+\lambda I$ places under every eigenvalue of G_t. If $G_t \approx I$, then $S_t^* \approx C_t$. Ordinary linear attention effectively makes precisely this approximation: it stores C_t itself as the state.

2 Why G_t is "curvature"

To understand G_t, fix a unit eigenvector $u \in \mathbb{R}^{d_k}$ with

$$G_t u = \gamma u, \qquad \|u\| = 1.$$

The eigenvalue γ turns out to carry two apparently different meanings. The fact that they are one and the same number is the whole story of the section.

Meaning 1: γ is the total overlap of the keys with u. Since $\|u\| = 1$,

$$\gamma = u^\top G_t u = u^\top \Big(\sum_i k_i k_i^\top + \lambda I \Big) u = \sum_i (u^\top k_i)^2 + \lambda.$$

Each term $(u^\top k_i)^2$ is the squared overlap of the direction u with the key k_i, so γ is the total accumulated (squared) overlap of the entire history with u, sitting above a floor λ. A large γ means many keys have piled up along u; a small γ means the history has barely visited that direction.

Meaning 2: γ is the curvature of the loss along the u direction. Perturb the state along the rank-one direction $\Delta S = a u^\top$, where $a \in \mathbb{R}^{d_v}$ is a unit value-space vector. This changes only how the map treats the u-component of a key,

$$(a u^\top) x = a \langle u, x \rangle,$$

sending that component's output along a. Along the straight line $S(\varepsilon) = S + \varepsilon a u^\top$,

$$\begin{aligned} \frac{d^2}{d\varepsilon^2} L_t(S + \varepsilon a u^\top) &= \|a\|^2 \Big(\sum_i (u^\top k_i)^2 + \lambda \|u\|^2 \Big) \\ &= u^\top G_t u = \gamma. \end{aligned}$$

So the loss surface bends with second derivative exactly γ as the state moves in the u direction. (The coordinate-free version of this computation is that the Hessian of L_t, as a linear operator on matrix perturbations, is

$$\boxed{H_t[\Delta S] = \Delta S \, G_t,} \tag{1}$$

obtained by differentiating the gradient: $\nabla L_t(S + \Delta S) - \nabla L_t(S) = \Delta S \, G_t$.)

Why they coincide. Both meanings are literally the same sum $\sum_i (u^\top k_i)^2 + \lambda$, and the reason is intuitive: the loss bends sharply in exactly those directions where many keys have accumulated, because moving S there changes many past predictions at once. This is what earns G_t the name *curvature*, and it makes G_t a data-induced metric on key space, $\|u\|_{G_t}^2 = u^\top G_t u$ (positive-definite when $\lambda > 0$, only semidefinite without ridge).

Consequence: steps should be scaled by curvature. Suppose we want to descend this loss surface toward its minimum. The right move is *not* the same length in every direction. In a direction where the surface is nearly flat (small γ) we can travel far; in a direction where it bends sharply (large γ) the surface falls away and turns back quickly, so a large step overshoots and we must move cautiously. The natural fix is to scale the step along u by $1/\gamma$: shorter steps in high-curvature directions, longer steps in flat ones. Rescaling each eigendirection by the inverse of its curvature is exactly what *preconditioning* does.

3 DeltaNet as online gradient descent

The ordinary DeltaNet update (Section 2.3 and paper Eq. (4)) is

$$S_t = S_{t-1} - \beta_t (S_{t-1} k_t - v_t) k_t^\top. \tag{P4 / paper Eq. 4}$$

But for the current-token loss

$$\ell_t(S) = \frac{1}{2} \|S k_t - v_t\|^2,$$

we have

$$\nabla \ell_t(S) = (S k_t - v_t) k_t^\top.$$

Therefore

$$\boxed{S_t = S_{t-1} - \beta_t \nabla \ell_t(S_{t-1}),}$$

so the delta rule is literally one **online** SGD step, with β_t playing the role of the online learning rate.

DeltaNet takes a uniform step. The scalar β_t multiplies the whole gradient, so it applies the *same* step length in every key-space direction. But we just argued that the right step should be scaled by $1/\gamma_j$, which differs across directions when the curvatures γ_j differ. A single scalar must be conservative enough for the sharpest-curvature direction and is therefore too timid in the flat ones. DeltaNet has no per-direction knowledge to do better, because

$$\nabla \ell_t(S_{t-1}) = (S_{t-1} k_t - v_t) k_t^\top$$

depends only on the *current* pair (k_t, v_t): it points along the single key direction k_t and contains no trace of the earlier keys $k_1, \ldots, k_{t-1}$, hence none of the geometry G_t.

The uniform step is optimal, but for a smaller problem. Put differently, DeltaNet is solving the right kind of step for the *wrong* objective. Its gradient is that of the instantaneous loss $\ell_t(S) = \frac{1}{2}\|Sk_t - v_t\|^2$, whose only data is the current token. On that one-point problem, with no accumulated curvature to consult, a scalar step is all the information there is. The objective we actually care about, though, is the *cumulative* regression L_t over the whole history—and solving that requires its curvature G_t.

With the curvature, one offline step suffices. Feed the cumulative curvature into the step and the picture changes completely. On the fixed cumulative quadratic L_t, the curvature-corrected (Newton) step is $-H_t^{-1}[\nabla L_t(S)]$; recalling from (1) that $H_t^{-1}[\Delta S] = \Delta S\, G_t^{-1}$,

$$\begin{aligned} S_{\text{new}} &= S - \nabla L_t(S)\, G_t^{-1} \\ &= S - (SG_t - C_t)G_t^{-1} \\ &= C_t G_t^{-1} = S_t^*. \end{aligned}$$

So exact curvature moves the state the right amount in every eigendirection and lands on the optimum S_t^* in a single step. (One step because the loss is quadratic: its Hessian is constant, so the local Newton step is globally exact.) In the eigenbasis this is just the per-direction rescaling promised above,

$$G_t^{-1}u_j = \frac{1}{\gamma_j}u_j, \qquad \boxed{\text{effective step along } u_j \;\propto\; \frac{1}{\gamma_j},}$$

with high-curvature, heavily occupied directions receiving smaller updates and flat directions larger ones. This is the *offline* solve; the next section shows how the paper realizes the same correction, once offline and once online.

4 Matching this viewpoint to Section 3 of the paper

The paper defines

$$P_t := G_t^{-1}, \qquad S_t^* := C_t P_t.$$

It realizes the same curvature correction in two ways: an *offline* form, which applies the preconditioner at read time on top of ordinary linear attention (ATQ), and an *online* form, which folds the preconditioner into the DeltaNet write so that the recurrence itself tracks the solution token by token (ATK). The two look different but compute the identical map $S_t^* = C_t P_t$; Sherman–Morrison is the bridge that proves the online recurrence reproduces the offline solve exactly at every step.

Linear attention: apply the preconditioner to the query (ATQ). Linear attention already maintains

$$C_t = C_{t-1} + v_t k_t^\top.$$

The exact regression output is

$$o_t = (C_t P_t)q_t = C_t(P_t q_t) = C_t \widetilde{q}_t. \qquad \text{(P3 / paper Eq. 3)}$$

Thus ordinary linear attention amounts to treating G_t as approximately I, while ATQ corrects this by transforming the query by the inverse curvature P_t.

DeltaNet: apply the preconditioner to the write key (ATK). For an exact online update, the paper defines the normalized inverse-Gram preconditioner

$$P_{t-1}^{\text{norm}} = \frac{P_{t-1}}{1 + k_t^\top P_{t-1} k_t}, \qquad \widetilde{k}_t = P_{t-1}^{\text{norm}} k_t,$$

and replaces the ordinary write key by $\widetilde{k}_t$:

$$\boxed{S_t = S_{t-1} + (v_t - S_{t-1}k_t)\widetilde{k}_t^\top.} \qquad \text{(P5 / paper Eq. 5)}$$

This is the paper's *apply-to-key* (ATK) realization, and the normalization is where Sherman–Morrison enters.

Sherman–Morrison. For an invertible M and vectors x, y, the inverse of a rank-one update is

$$(M + xy^\top)^{-1} = M^{-1} - \frac{M^{-1}x\, y^\top M^{-1}}{1 + y^\top M^{-1} x}.$$

The point is that once M^{-1} is known, adding a single rank-one term does *not* require a fresh $O(d_k^3)$ inversion: the correction is a rank-one matrix computed in $O(d_k^2)$, and the scalar denominator $1 + y^\top M^{-1}x$ is exactly what keeps $(M + xy^\top)^{-1}(M + xy^\top) = I$. Here each token appends $k_t k_t^\top$ to the Gram, so with $M = G_{t-1}$, $x = y = k_t$,

$$P_t = G_t^{-1} = (G_{t-1} + k_t k_t^\top)^{-1} = P_{t-1} - \frac{P_{t-1}k_t k_t^\top P_{t-1}}{1 + k_t^\top P_{t-1} k_t}.$$

This is why $P_{t-1}^{\text{norm}} = P_{t-1}/(1 + k_t^\top P_{t-1} k_t)$ carries the denominator: it is the Sherman–Morrison normalizer for the incremental update, and it is precisely this scalar that makes the online write $\widetilde{k}_t = P_{t-1}^{\text{norm}} k_t$ reproduce the offline solve exactly (rather than merely approximately). Substituting the rank-one P_t above into $S_t = C_t P_t$ and simplifying collapses to the ATK recurrence (P5); Theorem 3.1 proves that, initialized with $S_0 = 0$ and $P_0 = \lambda^{-1} I$ (i.e. $G_0 = \lambda I$),

$$S_t = C_t P_t$$

for every t. Thus exact preconditioned DeltaNet recursively tracks the same exact least-squares solution as exact preconditioned linear attention.

5 Why approximate the preconditioner?

The exact inverse Gram is expensive and introduces sequential dependence. Section 3.3 therefore keeps only its diagonal information:

$$A_t = A_{t-1} + k_t \odot k_t, \qquad P_t \approx \mathrm{Diag}(A_t)^{-1}. \qquad \text{(P7 / paper Eq. 7)}$$

Coordinate j therefore tracks roughly

$$(A_t)_j = \sum_{i \le t} k_{i,j}^2,$$

i.e. the diagonal $\mathrm{diag}(G_t)$ (up to the ridge offset λ): the accumulated second moment—and hence approximate curvature—along coordinate j. The resulting inverse rescales the update coordinatewise, much like a diagonal second-order optimizer. Note that it inverts the second moment directly, $1/(A_t)_j$, rather than its square root as in Adagrad/RMSProp: the intent here is to approximate the diagonal of the Hessian G_t, not to normalize gradients. The paper's Figure 1 provides empirical motivation: the key-Gram eigenvectors are sufficiently axis-aligned that a diagonal approximation retains useful curvature information.

Finally, the practical PDN recurrence in Section 3.5, Eq. (9), does not use the raw reciprocal directly. It introduces learned decay/gain in the second-moment state and a bounded transformation B_t for trainability and stability, then writes

$$\widetilde{k}_t = B_t \odot k_t, \qquad S_t = S_{t-1} - \beta_t (S_{t-1} k_t - v_t) \widetilde{k}_t^\top.$$

So the conceptual progression is

keys seen so far
$\Downarrow$
$G_t = \sum_i k_i k_i^\top + \lambda I$
$\Downarrow$
curvature / geometry of the regression problem
$\Downarrow$
G_t^{-1} (or a diagonal approximation)
$\Downarrow$
direction-dependent rescaling of the DeltaNet update.

Takeaway. Vanilla DeltaNet is already an online learning algorithm: it takes one first-order SGD step per key–value pair. The improvement proposed here is not "do regression instead of the delta rule," but rather *solve the same regression problem with information about its geometry.* The key Gram matrix is simultaneously (i) the accumulated second moment of the keys, (ii) the Hessian on the key dimension of the least-squares objective, and (iii) a data-induced metric describing which key directions are heavily occupied. Preconditioning approximately inverts this geometry, turning the scalar online learning rate of the delta rule into a direction-dependent one.