Project Notes & Findings

Research Log

Started: Dec 2025, Last update: Jan 2026

1 Introduction

The first objective of this project is to challenge our understanding of regularization in continual learning, and more specifically explain why and when quadratic regularizers are effective and when they fail.

2 Background on quadratic approximations

The most widely used regularizers are quadratic functions in θ, which typically consist in a Mahlanobis distance from an anchor point, usually chosen to be the model parameters at the end of the previous observation cycle $\theta^\star_{t-1}$. In formula:

$$g_{t-1}(\theta) = (\theta - \theta^\star_{t-1})^\top Q_{t-1}(\theta - \theta^\star_{t-1}), \tag{1}$$

where Q_{t-1} is a positive semi-definite matrix encoding the metric. The first use of quadratic regularization was in Elastic Weight Consolidation (EWC), where Q_{t-1} is the diagonal of the Fisher Information Matrix. The regularizer can be interpreted as a replacement of the previous tasks' cumulative loss, $\ell_{1:t-1}(\theta) = \sum_{i=1}^{t-1} \ell_i(\theta)$, where ℓ_i is the loss on task i.

The choice of approximation eq. (1) may seem too restrictive and inaccurate for neural networks. The formula can be easily derived from the second-order Taylor expansion of the loss function $\mathcal{T}_2(\theta; \ell_{1:t-1}, \theta^\star_{t-1})$.

$$\mathcal{T}_2(\theta; \ell_{1:t-1}, \theta^\star_{t-1}) = \nabla \ell_{1:t-1}(\theta^\star_{t-1})^\top(\theta - \theta^\star_{t-1}) + \tfrac{1}{2}(\theta - \theta^\star_{t-1})^\top \nabla^2 \ell_{1:t-1}(\theta^\star_{t-1})(\theta - \theta^\star_{t-1}) + \text{cnst}.$$

When $\theta^\star_{t-1}$ is a local minimum, the first term vanishes, and one recovers the quadratic form with $Q_{t-1} = \nabla^2 \ell_{1:t-1}(\theta^\star_{t-1})$ (up to a constant term). Moreover, when the model fits the data well (i.e., the error is close to zero), the Hessian is dominated by its Generalized Gauss-Newton (GGN) component:

$$\nabla^2_\theta \ell_{1:t-1} = \underbrace{\sum_{i=1}^{t-1} J_i^\top H_i \, J_i}_{\text{GGN}} + \underbrace{\sum_{i=1}^{t-1} \sum_{k=1}^{C} \frac{\partial \ell_i}{\partial f_{k,i}} \nabla^2_\theta f_{k,i}}_{\text{Model Curvature}}$$

because the so-called *residuals* $\frac{\partial \ell_i}{\partial f_{k,i}}$ are small near the optimum. Here, J_i is the Jacobian of the model outputs with respect to parameters, and H_i is the Hessian of the loss with respect to model outputs.

For negative log-likelihood objectives in the exponential family, the GGN coincides exactly with the Fisher Information Matrix:

$$F = \mathbb{E}_{x,y \sim p_{data}} \left[\nabla_\theta \log p_y(x) \nabla_\theta \log p_y(x)^\top \right],$$

where, in the case of a multi-class classification problem, $p_y(x)$ is the predicted probability of class y given sample x, obtained by applying the softmax to the model logits $f_\theta(x)$. Focusing on the classification case, for C classes, this is computed as:

$$F_{t-1} = \frac{1}{t-1} \sum_{i=1}^{t-1} \sum_{k=1}^{C} p_c(x_i) \nabla_\theta \log p_c(x_i) \nabla_\theta \log p_c(x_i)^\top$$

The Fisher might be estimated more efficiently using only the observed class, leading to the so-called *Empirical Fisher* approximation:

$$\hat{F}_{t-1} = \frac{1}{t-1} \sum_{i=1}^{t-1} \nabla_\theta \log p_{y_i}(x_i) \nabla_\theta \log p_{y_i}(x_i)^\top$$

Note that for a single input x_i, the empirical Fisher matrix is of rank one. Also note that in the limit of zero error and maximal confidence, both the empirical Fisher and the true Fisher matrices vanish in the case of multi-class classification. The same is not true for the regression case, where the true Fisher never vanishes.

The Fisher offers two distinct advantages over the exact Hessian: (1) it is guaranteed to be positive-semi-definite, ensuring convexity and non-negativity and (2) it can be computed from first derivatives only, which are cheaper than second derivatives. For very large models, materializing the full $P \times P$ matrix in memory may be infeasible (where P is the size of the model); thus, practical implementations rely on diagonal or block-diagonal approximations.

The derivation from Taylor expansion exposes a fundamental limitation of all approximations in the form of eq. (1): like the expansion itself, they are accurate only within a local neighborhood of the center $\theta^\star_{t-1}$. In practice, regularization methods are often used "improperly" allowing the model to drift far outside the region where the quadratic approximation remains valid (i.e., without a strictly enforced *trust region*). We believe that this fact explains the relative lower performance of regularization methods compared to other continual learning algorithms, and their sensitivity to hyperparameters influencing the local geometry—such as learning rate, batch size, and model capacity. Moreover, to minimize the cost, many methods update the metric Q_{t-1} using only the most recent observation x_t, exploiting the additivity of the Fisher/Hessian. Therefore, the regularizer becomes:

$$g_{t-1}(\theta) = \sum_{i=1}^{t-1} (\theta - \theta^\star_i)^\top Q_i(\theta^\star_i)(\theta - \theta^\star_i)$$

While this reduces the complexity of the regularizer update from $O(t)$ to $O(1)$, it introduces approximation errors as it further increases the distance between the current parameters θ and the approximation anchor points $\theta^\star_i$ for $i < t-1$.

Our goal in this short paper is to prove this hypothesis, and quantifying the effect of each source of error on the final performance of quadratic regularizers.

3 Sources of error

Based on the above exposition, we identify the following sources of error:

- **The refresh rate of the approximation.** The curvature matrix Q_i is computed at the minimum of task i. When the model drifts far from this point during training on subsequent tasks, the approximation becomes inaccurate. We call this choice the *'accumulate'* option, and compare it with a strategy where the curvature is recomputed at every task end (the *'reset'* option).
- **The curvature matrix choice.** We compare different choices for the curvature matrix Q_i, including the Hessian, the Generalized Gauss-Newton, and the Empirical Fisher Information Matrix.
- **The optimization implicit bias.** We investigate the effect of the choice of training trajectory on the approximation accuracy. In particular, we compare a standard gradient descent trajectory (called *'sequential'*) with one that adds the regularization term to the objective (called *'regularized'*) and finally with a trajectory that employs *'replay'* of all past data.

Metrics We compute the regularizer on a subset of the task data, namely a random 10% of the training set. For each sample, we track the following metrics during training (at each step):

- **Approximation Error.** We measure the relative error between the true loss increase and its quadratic approximation at each training step:

  $$\kappa(x, \theta) = \ell(x, \theta) - g(x, \theta) - \ell(x, \theta^{\star}_{t-1})$$

  where $g(x, \theta)$ is defined as in eq. (1), and the curvature matrix used is computed on x only. Additionally, we track the gradient error in norm and angle:

  $$\gamma(x, \theta) = \|\nabla_\theta \ell(x, \theta) - \nabla_\theta g(x, \theta)\|_2 \quad \text{and} \quad \alpha(x, \theta) = \cos\left(\nabla_\theta \ell(x, \theta), \nabla_\theta g(x, \theta)\right).$$

- **Task Performance.** We track the accuracy on the current task, the average accuracy across all previous tasks, and the average accuracy across regularized samples.
- **Several landscape metrics.** We additionally track the sharpness, rank and trace of the curvature matrix for the current task and for the regularized samples.

Initial results. We executed some initial experiments on a small model (one hidden layer MLP with 502 parameters) on a sequence of 9 binary classification tasks in 2D.

Table 1: Impact of Curvature Refreshing on Accuracy and Approximation Fidelity

Strategy	All Acc. ($\uparrow$)	Reg. Acc. ($\uparrow$)	κ Loss ($\downarrow$)	κ Grad ($\downarrow$)
		Final Step		
Accumulate	0.688 ± 0.053	0.654 ± 0.051	3.382 ± 1.015	6.354 ± 0.334
Refresh	$\mathbf{0.742} \pm 0.037$	$\mathbf{0.725} \pm 0.025$	$\mathbf{0.056} \pm 0.064$	$\mathbf{5.522} \pm 0.593$
		Trajectory Average		
Accumulate	0.668 ± 0.053	0.662 ± 0.026	2.845 ± 0.872	6.379 ± 0.571
Refresh	$\mathbf{0.710} \pm 0.069$	$\mathbf{0.705} \pm 0.021$	$\mathbf{0.407} \pm 0.134$	$\mathbf{5.825} \pm 0.392$

The refresh rate of the approximation. In table 1, we compare the 'accumulate' and 'refresh' strategies for curvature computation using the True Fisher Information Matrix. The results indicate that refreshing the curvature at each task end significantly improves both the overall accuracy and the fidelity of the quadratic approximation, as evidenced by lower κ loss and gradient errors. These findings suggest that maintaining an up-to-date curvature matrix is crucial for effective regularization in continual learning scenarios.

The objective of this project is to study the geometric properties of the loss landscape in deep neural networks and leverage these properties for better continual learning and regularization. We specifically focus on the **Fisher Information Matrix (FIM)** as a proxy for local curvature.

4 Fisher Information: Theoretical Foundations

We utilize the "True Fisher" for our curvature estimates. We have mathematically verified that the definition based on the expectation of log-probability gradients is equivalent to the Jacobian-based form used in implementation.

4.1 Equivalence Derivation

Given a model with logits $\boldsymbol{z}$ and probabilities $\boldsymbol{p} = \text{softmax}(\boldsymbol{z})$, the FIM F can be expressed in two ways:

1. **Probability Form:** $F = \sum_c p_c(\nabla_\theta \log p_c)(\nabla_\theta \log p_c)^\top$
2. **Logit Form:** $F = J^\top(\text{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top)J$

Where J is the Jacobian of the logits with respect to parameters. The equivalence holds because the term $(\text{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top)$ is the Fisher Information of the softmax distribution itself.

5 Spectral Analysis and Regularization

5.1 Numerical Rank

Empirical observation shows that the FIM is significantly low-rank. We define the *Numerical Rank* as the number of eigenvalues λ_i such that $\lambda_i > \epsilon \cdot \lambda_{\max}$. This allows us to isolate the "informative" directions of the parameter space.

5.2 Soft Regularization and Scaling

To maintain a balance between task loss and regularization, we utilize a spectral override where the basis V is scaled by the sharpness:

$$V_{scaled} = V_k \cdot \sqrt{\lambda_{\max}} \tag{2}$$

This ensures that the quadratic penalty $\|\Delta^\top V_{scaled}\|^2$ matches the actual curvature magnitude $\lambda_{\max}$ of the task, preventing the regularizer from being disproportionately weak.

6 Hard Regularization: Gradient Projection

Instead of a penalty, we implemented a **Projected Gradient Descent** approach to enforce constraints.

6.1 The Sequential Projection Failure

A critical finding was that sequential projection onto non-orthogonal subspaces S_1, S_2 leads to "leakage." If P_1 and P_2 are projection operators:

$$v^\top (P_2 P_1 \theta) \neq 0 \quad \text{if} \quad v \in S_1, u \in S_2, v^\top u \neq 0 \tag{3}$$

The act of satisfying the second constraint re-introduces violations of the first.

6.2 Global Orthogonal Basis (QR)

To solve this, we compute a **Global Basis** Q using the QR decomposition of the union of all task subspaces. The projection is then performed in a single step:

$$\Delta_{safe} = \Delta_{update} - Q(Q^\top \Delta_{update}) \tag{4}$$

This ensures the update lies in the intersection of the null spaces of all previous tasks.

7 Empirical Observations

7.1 The Rank-Cut Sanity Check

In an experiment with a model of $D = 502$ parameters, we applied a hard rank cut of $k = 1000$.

- **Expectation:** The model should be entirely frozen as the projection matrix $P = I - QQ^\top$ becomes the zero matrix.
- **Observation:** Accuracy fluctuates during training batches.
- **Explanation:** This is attributed to *sampling noise* (batch variance) rather than weight updates. If parameters are logged, the L_2 distance from the start of the task remains effectively zero ($< 10^{-7}$).