Enhancing Trustworthiness Notes

Jayin Khanna

June 2026

1 The Problem Statement

1.1 A Motivating Example

Suppose a legal services firm fine-tunes LLaMA on 50,000 internally curated legal question–answer pairs. The dataset is entirely benign: there are no harmful instructions, adversarial prompts, or toxic examples; only domain-specific legal reasoning. After supervised fine-tuning (SFT), the model's perplexity on legal queries decreases substantially, indicating improved performance on the target domain. Encouraged by these results, the firm deploys the model.

However, users soon begin to notice several unexpected behaviors:

- The model confidently states that certain jurisdictions permit actions that are, in reality, prohibited.
- It occasionally employs subtly gendered language when describing professional roles.
- Its reasoning often appears legally sound but contains factual inaccuracies in specific situations.

Importantly, none of these problematic behaviors were explicitly present in the training dataset. The data itself was clean and carefully curated.

> The central challenge is that *fine-tuning can unintentionally distort desirable behaviors learned during pretraining*, even when the fine-tuning dataset contains no harmful content.

This phenomenon is not merely hypothetical. **Qi et al. (2024)** demonstrated empirically that fine-tuning on fully benign datasets can significantly degrade a model's truthfulness, increase stereotypical bias scores, and reduce machine ethics scores relative to the original pre-trained model. In other words, optimizing the standard cross-entropy objective for a downstream task may inadvertently overwrite portions of the alignment knowledge acquired during large-scale pretraining.

1.2 Existing Approaches

Once such trustworthiness degradations are observed, several standard remedies are available.

Option A: RLHF / DPO Post-Processing

One possibility is to perform an additional alignment stage using preference-based learning methods such as Reinforcement Learning from Human Feedback (RLHF) or Direct Preference

Optimization (DPO). In this setting, human annotators construct preference pairs consisting of preferred and rejected responses, and the model is optimized accordingly.

While effective, this approach suffers from several drawbacks:

1. Constructing high-quality preference pairs for specialized domains requires expensive human annotation by domain experts.
2. RLHF may negatively impact performance on the original task if the KL regularization coefficient is not carefully calibrated.
3. Training costs remain substantial. For example, DPO training for a 7B-parameter model can require approximately 17 hours per run.

Option B: Data Filtering Before Fine-Tuning

Another common strategy is to filter potentially harmful, toxic, or biased examples before the supervised fine-tuning stage.

This approach also faces important limitations:

1. In many practical settings, the dataset is already benign, leaving little obvious content to remove.
2. Existing filtering systems often rely on surface-level heuristics such as keyword matching and therefore fail to detect subtle semantic influences that emerge only after optimization.

Option C: Post-Hoc Data Debugging and Targeted Repair

The approach proposed in this paper follows a fundamentally different philosophy.

Rather than modifying the training pipeline before fine-tuning or introducing a costly alignment stage afterward, the method operates directly on the already fine-tuned model.

The procedure consists of three steps:

1. Identify the specific training samples that are most responsible for the observed trustworthiness degradation.
2. Select a small and diverse subset of these influential samples.
3. Perform a minimal targeted gradient-ascent update that reduces the influence of those samples while leaving the remainder of the model largely unchanged.

Post-Hoc Data Debugging refers to the process of identifying and correcting harmful influences of individual training examples after model training has already been completed.

This strategy offers several advantages:

- No retraining from scratch.
- No construction of preference datasets.
- No additional RLHF or DPO optimization stage.
- Runtime measured in minutes rather than hours.

1.3 The Core Question

The paper addresses the following fundamental question:

Can we identify the small subset of training examples responsible for trustworthiness degradation and selectively remove their influence, while preserving the task-specific capabilities gained during

fine-tuning?

The remainder of the paper develops a principled framework for answering this question through influence estimation, sample selection, and targeted model repair.

2 Formal Problem Setup

Let me introduce notation cleanly so later everything slots in.

Given.
You have:

(1) A model $M(\theta)$ — an LLM with parameters

$$\theta \in \mathbb{R}^d,$$

where d is very large (billions).

(2) A training dataset

$$\mathcal{D}_{\text{train}} = \{z_i\}_{i=1}^n,$$

where each

$$z_i = (x_i, y_i)$$

is a prompt-response pair, drawn from some distribution $P_{\mathcal{D}}$.

(3) θ^{post} — the parameters after SFT on $\mathcal{D}_{\text{train}}$.
This is your starting point. The SFT is already done and you can't undo it cheaply.

You also have K trustworthiness dimensions (truthfulness, stereotypical bias, machine ethics).

For each dimension j, you have:

(a) An evaluation dataset

$$\mathcal{D}^j_{\text{trust}} = \{v_i\},$$

where each

$$v_i = (m_i, p_i, o_i),$$

a prompt m_i, a proponent (good response) p_i, and an opponent (bad response) o_i.

(b) A metric

$$F^j(v; \theta),$$

measuring how well θ handles dimension j on sample v.
Lower is better.

(c) A separate metric

$$T(z; \theta),$$

for downstream task performance (operationalized as perplexity on $\mathcal{D}_{\text{train}}$).
Lower is better.

Objective.
Find new parameters θ (starting from θ^{post}, within a compute budget) such that

$$\mathbb{E}_{v \sim P^k_{\text{trust}}}\left[F^k(v; \theta)\right] \leq \mathbb{E}_{v \sim P^k_{\text{trust}}}\left[F^k(v; \theta^{\text{post}})\right]. \tag{1}$$

and

$$\left|\mathbb{E}_{z \sim P_{\mathcal{D}}}[T(z; \theta)] - \mathbb{E}_{z \sim P_{\mathcal{D}}}\left[T(z; \theta^{\text{post}})\right]\right| \leq \epsilon. \tag{2}$$

In words: improve the trust metric without meaningfully degrading perplexity, and do it cheaply.

The two constraints are genuinely in tension. Any parameter update that improves trust risks hurting perplexity. The entire methodological challenge is navigating this tradeoff.

3 The Implicit Function Theorem Applied to Optimality Conditions

3.1 Setup

You have n training points $z_1, \ldots, z_n$, where $z_i = (x_i, y_i)$. You train a model by minimizing empirical risk:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(z_i, \theta). \tag{3}$$

Call

$$R(\theta) = \frac{1}{n} \sum_{i} \mathcal{L}(z_i, \theta) \tag{4}$$

the empirical risk. So

$$\hat{\theta} = \arg\min_{\theta} R(\theta). \tag{5}$$

The question: Without retraining, what happens to $\hat{\theta}$ if we remove one training sample z?

Removing z from a dataset of n points is the same as giving it weight $-\frac{1}{n}$ on top of the uniform weighting. The paper uses a cleaner equivalent: instead of removing z, consider *upweighting* z by a small amount ϵ, giving the perturbed objective

$$R_\epsilon(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(z_i, \theta) + \epsilon \cdot \mathcal{L}(z, \theta). \tag{6}$$

Let $\hat{\theta}_{\epsilon,z}$ be the minimizer of this perturbed objective. We want to know how $\hat{\theta}_{\epsilon,z}$ moves as ϵ changes — specifically,

$$\left. \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \right|_{\epsilon=0}. \tag{7}$$

Then removal corresponds to $\epsilon = -\frac{1}{n}$, so

$$\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n} \left. \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \right|_{\epsilon=0}. \tag{8}$$

This is the linearization step. It's valid because $\epsilon = -\frac{1}{n}$ is small when n is large. The whole derivation is just computing that derivative.

3.2 The Original Objective

You have n training points, each with equal weight $\frac{1}{n}$. The empirical risk is

$$R(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(z_i, \theta). \tag{9}$$

You can think of this as a weighted sum where every point z_i gets weight $\frac{1}{n}$. Rewrite it explicitly as

$$R(\theta) = \sum_{i=1}^{n} w_i \cdot \mathcal{L}(z_i, \theta), \qquad w_i = \frac{1}{n} \text{ for all } i. \tag{10}$$

The minimizer is

$$\hat{\theta} = \arg\min_{\theta} R(\theta). \tag{11}$$

Suppose $z = z_k$ for some specific index k. Removing it and retraining means solving

$$\hat{\theta}_{-z} = \arg\min_{\theta} \sum_{i \neq k} \frac{1}{n-1} \mathcal{L}(z_i, \theta). \tag{12}$$

The remaining $n - 1$ points each get weight $\frac{1}{n-1}$.

But this changes two things simultaneously:

(a) z_k disappears.
(b) Every other point's weight changes from $\frac{1}{n}$ to $\frac{1}{n-1}$.

That makes the analysis messy — you can't isolate the effect of z_k alone.

3.3 The Cleaner Equivalent Formulation

Instead of the above, consider a different operation: keep all n points, but change only z_k's weight. Set z_k's weight to zero, keep everyone else at $\frac{1}{n}$:

$$\tilde{R}(\theta) = \sum_{i \neq k} \frac{1}{n} \mathcal{L}(z_i, \theta) + 0 \cdot \mathcal{L}(z_k, \theta). \tag{13}$$

This is equivalent to the original $R(\theta)$ minus z_k's contribution:

$$\tilde{R}(\theta) = R(\theta) - \frac{1}{n} \mathcal{L}(z_k, \theta). \tag{14}$$

The minimizer of $\tilde{R}(\theta)$ is not exactly $\hat{\theta}_{-z}$ (because the remaining weights are $\frac{1}{n}$, not $\frac{1}{n-1}$), but for large n the difference between $\frac{1}{n}$ and $\frac{1}{n-1}$ is negligible. This is the approximation the paper implicitly makes — it's valid asymptotically as $n \to \infty$.

3.3.1 Now Generalize: Upweighting by ϵ

Instead of jumping straight to "set weight to zero," parameterize the weight of z_k continuously. Let ϵ be a small number you can tune:

$$R_\epsilon(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(z_i, \theta) + \epsilon \cdot \mathcal{L}(z_k, \theta). \tag{15}$$

At $\epsilon = 0$, you recover the original $R(\theta)$. The minimizer is $\hat{\theta}$.

At $\epsilon = -\frac{1}{n}$, z_k's effective weight becomes

$$\frac{1}{n} + \left(-\frac{1}{n}\right) = 0.$$

So

$$R_{-1/n}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(z_i, \theta) - \frac{1}{n}\mathcal{L}(z_k, \theta) = \sum_{i\neq k} \frac{1}{n}\mathcal{L}(z_i, \theta) = \tilde{R}(\theta). \tag{16}$$

Exactly the "removal" objective from above.

So the minimizer of $R_{-1/n}(\theta)$ is exactly $\hat{\theta}_{-z}$ (in the approximation where $\frac{1}{n} \approx \frac{1}{n-1}$).

3.3.2 Why Parameterize by ϵ at All?

Because now you have a smooth family of objectives $R_\epsilon(\theta)$, and the minimizer $\hat{\theta}_{\epsilon,z}$ is a smooth function of ϵ. You can differentiate:

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}. \tag{17}$$

This derivative tells you the rate at which the minimizer moves as you change z_k's weight, evaluated at the original weighting. This is the influence function.

Once you have this derivative, you recover the removal effect by the first-order Taylor expansion of $\hat{\theta}_{\epsilon,z}$ around $\epsilon = 0$:

$$\hat{\theta}_{\epsilon,z} \approx \hat{\theta} + \epsilon \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}. \tag{18}$$

Plug in $\epsilon = -\frac{1}{n}$:

$$\hat{\theta}_{-z} \approx \hat{\theta} - \frac{1}{n}\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}. \tag{19}$$

Rearranging,

$$\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n}\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}. \tag{20}$$

This is exactly the formula in the excerpt. You've now seen where it comes from — it's a first-order Taylor expansion of the minimizer path in ϵ, evaluated at the removal point $\epsilon = -\frac{1}{n}$.

$$\underbrace{\hat{\theta}_{-z} - \hat{\theta}}_{\text{what we want}} \approx \underbrace{-\frac{1}{n} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}}_{\text{what we can compute}}.$$

3.4 How to Compute

What We Need to Compute

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}$$

$\hat{\theta}_{\epsilon,z}$ is the minimizer of the perturbed objective

$$R_\epsilon(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(z_i, \theta) + \epsilon \cdot \mathcal{L}(z, \theta). \tag{21}$$

For each value of ϵ, $\hat{\theta}_{\epsilon,z}$ is defined implicitly as the point where the gradient of R_ϵ vanishes.

- There is no closed form for $\hat{\theta}_{\epsilon,z}$ as a function of ϵ.
- You therefore cannot differentiate an explicit formula.
- Instead, we compute the derivative through the optimality condition.

The tool for this is the implicit function theorem, applied to the optimality condition.

Step 1 — Write the Optimality Condition

Since $\hat{\theta}_{\epsilon,z}$ minimizes $R_\epsilon(\theta)$, it satisfies

$$\nabla_\theta R_\epsilon(\theta)\Big|_{\theta=\hat{\theta}_{\epsilon,z}} = 0. \tag{22}$$

Expanding,

$$\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \mathcal{L}(z_i, \hat{\theta}_{\epsilon,z}) + \epsilon \nabla_\theta \mathcal{L}(z, \hat{\theta}_{\epsilon,z}) = 0. \tag{23}$$

Step 2 — Differentiate with Respect to ϵ

Differentiate both sides of the optimality condition.

For the first term,

$$\frac{d}{d\epsilon}\left[\frac{1}{n}\sum_{i} \nabla_\theta \mathcal{L}(z_i, \hat{\theta}_{\epsilon,z})\right],$$

the dependence on ϵ comes through $\hat{\theta}_{\epsilon,z}$. Applying the chain rule gives

$$\frac{1}{n}\sum_i \nabla_\theta^2 \mathcal{L}(z_i, \hat{\theta}_{\epsilon,z}) \cdot \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}. \tag{24}$$

For the second term,

$$\frac{d}{d\epsilon}\left[\epsilon \cdot \nabla_\theta \mathcal{L}(z, \hat{\theta}_{\epsilon,z})\right],$$

apply the product rule:

$$\nabla_\theta \mathcal{L}(z, \hat{\theta}_{\epsilon,z}) + \epsilon \nabla_\theta^2 \mathcal{L}(z, \hat{\theta}_{\epsilon,z}) \cdot \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}. \tag{25}$$

Combining both terms and setting the derivative equal to zero gives

$$\left[\frac{1}{n}\sum_i \nabla_\theta^2 \mathcal{L}(z_i, \hat{\theta}_{\epsilon,z}) + \epsilon \nabla_\theta^2 \mathcal{L}(z, \hat{\theta}_{\epsilon,z})\right] \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} + \nabla_\theta \mathcal{L}(z, \hat{\theta}_{\epsilon,z}) = 0. \tag{26}$$

Step 3 — Evaluate at $\epsilon = 0$

At $\epsilon = 0$,

$$\hat{\theta}_{\epsilon,z}\Big|_{\epsilon=0} = \hat{\theta},$$

and the term

$$\epsilon \nabla_\theta^2 \mathcal{L}(z, \hat{\theta}_{\epsilon,z})$$

vanishes.
The previous equation becomes

$$\left[\frac{1}{n}\sum_i \nabla_\theta^2 \mathcal{L}(z_i, \hat{\theta})\right] \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\bigg|_{\epsilon=0} + \nabla_\theta \mathcal{L}(z, \hat{\theta}) = 0. \tag{27}$$

The bracketed term is exactly the Hessian of the empirical risk at $\hat{\theta}$. Define

$$H_{\hat{\theta}} = \frac{1}{n}\sum_i \nabla_\theta^2 \mathcal{L}(z_i, \hat{\theta}). \tag{28}$$

Therefore,

$$H_{\hat{\theta}} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} + \nabla_\theta \mathcal{L}(z, \hat{\theta}) = 0. \tag{29}$$

Step 4 — Solve for the Derivative

Rearranging,

$$H_{\hat{\theta}} \left. \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \right|_{\epsilon=0} = -\nabla_\theta \mathcal{L}(z, \hat{\theta}). \tag{30}$$

Multiplying both sides by $H_{\hat{\theta}}^{-1}$,

$$\left. \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z, \hat{\theta}). \tag{31}$$

Step 5 — Substitute Back

From the previous section,

$$\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n} \left. \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} \right|_{\epsilon=0}. \tag{32}$$

Substituting the derivative,

$$\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n} \left(-H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z, \hat{\theta}) \right). \tag{33}$$

Hence,

$$\boxed{\hat{\theta}_{-z} - \hat{\theta} \approx \frac{1}{n} H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z, \hat{\theta})} \tag{34}$$

This is the classical influence-function approximation for the parameter change induced by removing a single training example.

3.5 Back to paper*

We have two things.

Equation 1 — The Parameter Shift from Removing z_i.

$$\hat{\theta}_{-z_i} - \hat{\theta} \approx \frac{1}{n} H^{-1} \nabla_\theta \mathcal{L}(z_i, \hat{\theta}). \tag{35}$$

Equation 2 — First-Order Expansion of the Trust Metric. The trust metric $F^j(v; \theta)$ is a function of θ. Its first-order Taylor expansion around $\hat{\theta}$ is

$$F^j(v; \theta) \approx F^j(v; \hat{\theta}) + \nabla_\theta F^j(v; \hat{\theta})^\top (\theta - \hat{\theta}). \tag{36}$$

Substitute Equation 1 into Equation 2. Set

$$\theta = \hat{\theta}_{-z_i},$$

so that

$$\theta - \hat{\theta} = \hat{\theta}_{-z_i} - \hat{\theta}.$$

Substituting Equation 1 gives

$$F^j(v; \hat{\theta}_{-z_i}) \approx F^j(v; \hat{\theta}) + \nabla_\theta F^j(v; \hat{\theta})^\top \cdot \frac{1}{n} H^{-1} \nabla_\theta \mathcal{L}(z_i, \hat{\theta}). \tag{37}$$

Rearranging,

$$F^j(v; \hat{\theta}_{-z_i}) - F^j(v; \hat{\theta}) \approx \frac{1}{n} \nabla_\theta F^j(v; \hat{\theta})^\top H^{-1} \nabla_\theta \mathcal{L}(z_i, \hat{\theta}). \tag{38}$$

$$\boxed{F^j(v; \hat{\theta}_{-z_i}) - F^j(v; \hat{\theta}) \approx \frac{1}{n} \nabla_\theta F^j(v; \hat{\theta})^\top H^{-1} \nabla_\theta \mathcal{L}(z_i, \hat{\theta})} \tag{39}$$

The right-hand side is a scalar: the inner product of two vectors in $\mathbb{R}^d$.

(i)
$$\nabla_\theta F^j(v; \hat{\theta})$$
is the direction in parameter space along which the trust metric for evaluation sample v improves.

(ii)
$$H^{-1} \nabla_\theta \mathcal{L}(z_i, \hat{\theta})$$
is the direction the parameters move when z_i is removed, corrected for curvature through the inverse Hessian.

If these two vectors are aligned (positive inner product), then removing z_i improves the trust metric for v. In that case, z_i was hurting trust.
If they are anti-aligned (negative inner product), removing z_i makes trust worse. In that case, z_i was helping trust.

4 PBRF: From Influence Functions to Practical Model Repair

Recall the influence-function approximation

$$\hat{\theta}_{-z_i} - \hat{\theta} \approx \frac{1}{n} H^{-1} \nabla_\theta \mathcal{L}(z_i, \hat{\theta}). \tag{40}$$

The matrix H^{-1} is doing something very specific.

The Hessian H measures how curved the total training loss is in every direction of parameter space. Different directions have different curvatures: some directions are steep (high curvature), while others are relatively flat (low curvature).

The inverse Hessian rescales the gradient vector $\nabla_\theta \mathcal{L}(z_i, \hat{\theta})$ direction-by-direction according to this curvature.

- **High-curvature directions.**
  H^{-1} shrinks the update. Many other training samples have large gradients in these directions, so moving there would substantially increase their losses. Curvature resists movement.

- **Low-curvature directions.**
  H^{-1} amplifies the update. Few training samples care about these directions, so movement there causes little collateral damage.

Uniform Curvature Approximation.
If curvature were identical in every direction,

$$H = cI,$$

then

$$H^{-1} = \frac{1}{c} I.$$

Consequently,

$$H^{-1}\nabla_\theta \mathcal{L}(z_i, \hat{\theta}) = \frac{1}{c}\nabla_\theta \mathcal{L}(z_i, \hat{\theta}). \tag{41}$$

The curvature correction disappears and every direction receives exactly the same scaling.

This is precisely the assumption made by ordinary gradient ascent:

$$\theta \leftarrow \theta^{\text{post}} + \alpha \nabla_\theta \mathcal{L}(z_i, \theta^{\text{post}}). \tag{42}$$

The scalar α plays the role of a uniform approximation to

$$\frac{1}{nc}.$$

Gradient ascent therefore assumes the loss landscape is equally curved in every direction. Modern language models are far from this regime; curvature varies dramatically across parameter space.

4.1 The Tradeoff

The fundamental tradeoff is computational cost versus accuracy.

The direction

$$H^{-1}\nabla_\theta \mathcal{L}(z_i, \hat{\theta})$$

is the theoretically correct parameter update. It identifies directions in which the influence of z_i can be reduced while minimally disturbing other training samples.

Unfortunately, computing Hessian-inverse vector products for modern LLMs is prohibitively expensive.

By contrast,

$$\alpha \nabla_\theta \mathcal{L}(z_i, \theta^{\text{post}})$$

requires only a single backward pass and is therefore extremely cheap.

The drawback is that it completely ignores curvature.

Suppose the gradient $\nabla_\theta \mathcal{L}(z_i, \theta^{\text{post}})$ has a large component in a high-curvature direction. The true inverse-Hessian correction would heavily suppress that component. Ordinary gradient ascent does not. It steps with full magnitude in that direction, increasing the losses of many unrelated training examples as collateral damage.

This explains why simple gradient ascent methods often cause substantial increases in perplexity: they move aggressively in directions that the remainder of the training distribution strongly resists.

4.2 What "Don't Break the Model" Actually Means

Before defining an objective, we must formalize the notion of preserving model quality.

Several interpretations are possible.

Option 1: Stay Close in Parameter Space. Require

$$\|\theta - \theta^{\text{post}}\|^2$$

to remain small.

This criterion is weak. In highly overparameterized models, two parameter vectors can be very close while producing substantially different outputs, and vice versa.

Option 2: Stay Close in Output Space. Require the predictions on training inputs to remain close:

$$\mathcal{M}(x_i, \theta) \approx \mathcal{M}(x_i, \theta^{\text{post}}).$$

This criterion is more meaningful because perplexity depends on outputs rather than parameter values.

The remaining question is how to measure closeness in output space.

A simple squared-distance penalty treats all output directions equally. However, cross-entropy loss is not equally sensitive to all directions. Some output perturbations matter much more than others.

We therefore seek a geometry that weights output discrepancies according to their effect on the loss itself.

4.3 The Bregman Divergence

Let

$$g : \mathbb{R}^k \to \mathbb{R}$$

be a differentiable strictly convex function.

At a point u', the first-order Taylor approximation of g is

$$g(u') + \nabla g(u')^\top (u - u'). \tag{43}$$

Convexity implies

$$g(u) \geq g(u') + \nabla g(u')^\top (u - u') \qquad \forall u. \tag{44}$$

The difference between the function value and its tangent approximation is therefore nonnegative:

$$g(u) - g(u') - \nabla g(u')^\top (u - u') \geq 0. \tag{45}$$

Equality holds if and only if $u = u'$.

This gap defines the *Bregman divergence* generated by g:

$$D_g(u, u') = g(u) - g(u') - \nabla g(u')^\top (u - u'). \tag{46}$$

The Bregman divergence measures how much the true function deviates from its first-order approximation.

4.4 Assembling the PBRF Objective

We now have three ingredients.

Term A — Functional Anchor. Penalize deviations of the repaired model from the original model on the training distribution:

$$\frac{1}{N} \sum_{i=1}^{N} \Psi\Big(\mathcal{M}(x_i, \theta), \mathcal{M}(x_i, \theta^{\text{post}}); y_i\Big). \tag{47}$$

Term B — Ascent Pressure. Increase the loss on the identified detrimental subset S:

$$-\beta \sum_{(x,y)\in S} \mathcal{L}\big(\mathcal{M}(x, \theta), y\big). \tag{48}$$

Since the overall problem is a minimization, the negative sign causes the loss on S to be maximized.

Term C — Parameter Proximal Term. Penalize large parameter movements:

$$\frac{\lambda}{2} \|\theta - \theta^{\text{post}}\|^2. \tag{49}$$

The complete PBRF objective is therefore

$$\theta(\beta; S) = \arg\min_{\theta} \left[\frac{1}{N} \sum_{i=1}^{N} \Psi\Big(\mathcal{M}(x_i, \theta), \mathcal{M}(x_i, \theta^{\text{post}}); y_i\Big) - \beta \sum_{(x,y)\in S} \mathcal{L}\big(\mathcal{M}(x, \theta), y\big) + \frac{\lambda}{2} \|\theta - \theta^{\text{post}}\|^2 \right]. \tag{50}$$

Every term has a clear interpretation:

- The functional anchor preserves behavior on the training distribution.
- The ascent term removes the influence of detrimental samples.
- The proximal term prevents unnecessary parameter drift.

At $\beta = 0$, both the anchor term and the proximal term are minimized at

$$\theta = \theta^{\text{post}}.$$

Hence,

$$\theta(0; S) = \theta^{\text{post}},$$

which serves as the reference point for all subsequent analysis.