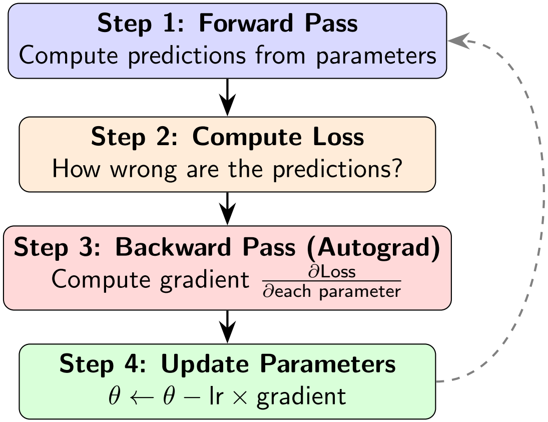
Why γ Doesn't Learn: Centered vs. Non-Centered MAP

How ML Training Works (4 Steps, Every Epoch)


Step 1: Forward Pass
Compute predictions from parameters
Step 2: Compute Loss
How wrong are the predictions?
Step 3: Backward Pass (Autograd)
Compute gradient $\frac{\partial \text{Loss}}{\partial \text{each parameter}}$
Step 4: Update Parameters
$\theta \leftarrow \theta - \text{lr} \times \text{gradient}$

Key rule: In Step 3, a parameter only gets a gradient from the loss if it was **used in Step 1** to compute the prediction. If a parameter isn't in the forward pass, autograd can't trace back to it.

The Hierarchical Model

$$\begin{tikzcd}
\gamma \arrow[d, "p(\lambda \mid \gamma)"] \\
\lambda \arrow[d, "p(Y \mid \lambda)"] \\
\boxed{Y}
\end{tikzcd}$$

ALADYN:

- γ = genetic effects on disease signatures (hyperparameter)

Centered Model: The Forward Pass

What the computer does each epoch:

1. Read λ from memory (it's a free `nn.Parameter`)

2. $\theta = \text{softmax}(\lambda)$ (mixing proportions)

3. $\pi = \theta \cdot \phi \cdot \kappa$ (disease probabilities)

4. $\text{Loss} = \text{NLL}(Y, \pi) + w \cdot (\lambda - \mu(\gamma))^\top \Omega^{-1} (\lambda - \mu(\gamma))$

Notice: γ appears **only in the prior term** (Step 4), never in the forward computation (Steps 1–3).

λ is read from memory $\to \theta \to \pi \to$ NLL.
The chain $Y \to \pi \to \theta \to \lambda$ does not pass through γ.

Reparameterized Model: The Forward Pass

What the computer does each epoch:

1. Read δ and γ from memory (both are `nn.Parameters`)

2. $\lambda = G \cdot \gamma + \delta$ (γ **enters the forward pass!**)

3. $\theta = \text{softmax}(\lambda)$ (mixing proportions)

4. $\pi = \theta \cdot \phi \cdot \kappa$ (disease probabilities)

5. $\text{Loss} = \text{NLL}(Y, \pi) + w \cdot \delta^\top \Omega^{-1} \delta$

Now: γ is **in the forward computation** (Step 2).
The chain is: $\gamma \to \lambda \to \theta \to \pi \to \text{NLL}$.

Autograd traces back through this chain, so γ gets a gradient from the data.

The Backward Pass: Why γ Gets Left Out

Centered model — backward pass computes:

$$\frac{\partial \text{Loss}}{\partial \lambda} = \underbrace{\frac{\partial \text{NLL}}{\partial \lambda}}_{\text{data signal (strong)}} + \underbrace{w \cdot \frac{\partial \text{prior}}{\partial \lambda}}_{\text{prior signal (weak)}}$$

$$\frac{\partial \text{Loss}}{\partial \gamma} = \underbrace{\overset{=\mathbf{0}}{\frac{\partial \text{NLL}}{\partial \gamma}}}_{\text{Not in forward pass!}} + \underbrace{w \cdot \frac{\partial \text{prior}}{\partial \gamma}}_{\text{prior only } (w=\text{1e-4})}$$

The Backward Pass: Why γ Gets Left Out

Centered model — backward pass computes:

$$\frac{\partial \text{Loss}}{\partial \lambda} = \underbrace{\frac{\partial \text{NLL}}{\partial \lambda}}_{\text{data signal (strong)}} + \underbrace{w \cdot \frac{\partial \text{prior}}{\partial \lambda}}_{\text{prior signal (weak)}}$$

$$\frac{\partial \text{Loss}}{\partial \gamma} = \underbrace{\overset{=\mathbf{0}}{\frac{\partial \text{NLL}}{\partial \gamma}}}_{\text{Not in forward pass!}} + \underbrace{w \cdot \frac{\partial \text{prior}}{\partial \gamma}}_{\text{prior only } (w\text{=1e-4})}$$

Reparam model — backward pass computes:

$$\frac{\partial \text{Loss}}{\partial \gamma} = \underbrace{\frac{\partial \text{NLL}}{\partial \pi} \cdot \frac{\partial \pi}{\partial \theta} \cdot \frac{\partial \theta}{\partial \lambda} \cdot \frac{\partial \lambda}{\partial \gamma}}_{\text{chain rule through forward pass!}} + 0$$

γ **is in the forward pass**, so the NLL gradient flows back to it.

Same Objective, Different Computation

Both models minimize the **same function**:

$$p(Y \mid \lambda) \cdot p(\lambda \mid \gamma)$$

Same Objective, Different Computation

Both models minimize the **same function**:

$$p(Y \mid \lambda) \cdot p(\lambda \mid \gamma)$$

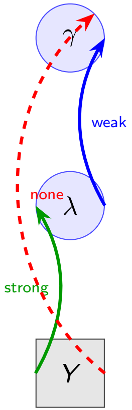
The difference is purely in **how the computer represents** λ:

	Centered	**Non-centered (Reparam)**
Free parameters	λ, γ	δ, γ
Forward pass	$\theta = \text{softmax}(\lambda)$	$\lambda = G\gamma + \delta$ $\theta = \text{softmax}(\lambda)$
NLL	$-\log p(Y \mid \lambda)$	$-\log p(Y \mid G\gamma + \delta)$
Prior	$(\lambda - G\gamma)^\top \Omega^{-1} (\lambda - G\gamma)$	$= \ \delta^\top \Omega^{-1} \delta$ (identical)
γ in NLL?	No	Yes
γ gets data grad?	No	Yes

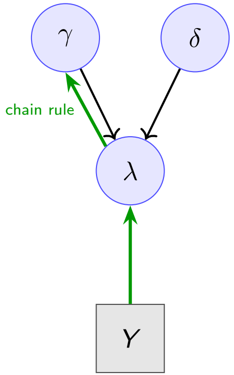
At any solution: $\delta^* = \lambda^* - G\gamma^*$. If the problem were convex, both would find the same optimum. Because it is **non-convex**, different gradient flows $\rightarrow$ different local optima.

The Gradient Flow Picture

Centered:



Non-centered:



Both γ and δ feed into λ in the forward pass $\rightarrow$ both get NLL gradients.
In centered, λ is a dead-end: gradients stop there and never reach γ through the NLL.

A Simple Analogy

Minimize $f(x) = (x - 3)^2$ with a preference that x be near μ:

Centered:

- Parameters: x and μ
- Loss $= (x - 3)^2 + w \cdot (x - \mu)^2$
- μ only appears in the penalty. If w is small, μ barely moves.

Non-centered:

- Parameters: z and μ, where $x = \mu + z$
- Loss $= (\mu + z - 3)^2 + w \cdot z^2$
- μ appears in the **main loss**. It gets a strong gradient.

Same answer ($x^* = 3$), but in the second form μ learns where x wants to be. In the first form, μ barely moves because w is tiny.

Why It Usually Doesn't Matter (for Prediction)

λ **compensates.**

For prediction, we:

1. Fix all population parameters $(\phi, \gamma, \psi, \kappa)$ from training
2. For each new individual: re-estimate λ_{new} (200 epochs)
3. λ_{new} has $K \times T$ free parameters — enough to absorb the signal

So even if $\gamma \approx 0$ (centered), λ_{new} can still fit the data well.

Why It Usually Doesn't Matter (for Prediction)

λ **compensates.**

For prediction, we:

1. Fix all population parameters $(\phi, \gamma, \psi, \kappa)$ from training
2. For each new individual: re-estimate λ_{new} (200 epochs)
3. λ_{new} has $K \times T$ free parameters — enough to absorb the signal

So even if $\gamma \approx 0$ (centered), λ_{new} can still fit the data well.

However, it does still matter because:

- Reparam initializes $\lambda = G\gamma + \delta$ (starting near the genetic prediction)
- Centered initializes λ near zero (no genetic information at start)
- Different starting points $\rightarrow$ different optimization paths $\rightarrow$ different final λ
- Different trained $\phi/\psi/\kappa$ between the two pipelines also contribute

Prediction AUC: Centered vs. Non-Centered

50,000 held-out patients, 100 bootstrap replicates, 28 diseases:

Metric	**Centered**	**Non-centered**	Δ	**Wins**
Static 10-year	0.622	0.654	$+0.032$	25/28
Dynamic 10-year	0.624	0.629	$+0.005$	18/28
Dynamic 1-year	0.765	0.883	$+0.118$	24/28

Non-centered wins on all 3 metrics.

- Largest advantage at short horizon (1-year: $+0.118$)
- Smallest advantage at dynamic 10-year ($+0.005$)
- **Caveat:** confounded by different population parameters (ϕ, ψ, κ) between the two training pipelines — not purely a parameterization effect

Trained Parameter Differences

	Centered	**Non-centered**
Mean $\|\gamma\|$	0.006	0.081
κ	2.93	4.52
γ correlation	0.37 (very different)	
ψ correlation	0.76 (moderately different)	
ϕ correlation	0.94 (similar)	

- ϕ (disease trajectories): nearly identical — both fit the data well
- ψ (signature assignments): moderately different — non-centered less stable
- γ (genetic effects): **centered** $\gamma \approx 0$ ($14\times$ smaller)

- The **prediction AUC difference is driven by all of these** (especially $\phi/\psi/\kappa$), not just γ

Summary

1. **ML training = forward $\rightarrow$ loss $\rightarrow$ backward $\rightarrow$ update.** A parameter only gets a data gradient if it was in the forward pass.
2. Both models optimize the same posterior: $p(Y|\lambda) \cdot p(\lambda|\gamma)$. The difference is whether you write $\lambda = G\gamma + \delta$ in the forward step.
3. **Centered** (original): λ is free, γ only in the prior $\rightarrow$ γ gets no data gradient.
4. **Non-centered** (reparam): $\lambda = G\gamma + \delta$, γ in forward pass $\rightarrow$ γ gets data gradient via chain rule.
5. Both are standard approaches. Centered is the default in Stan/lme4/PyMC. Non-centered is the standard alternative.
6. For prediction: non-centered wins on AUC (all 3 metrics), but confounded by different population parameters.