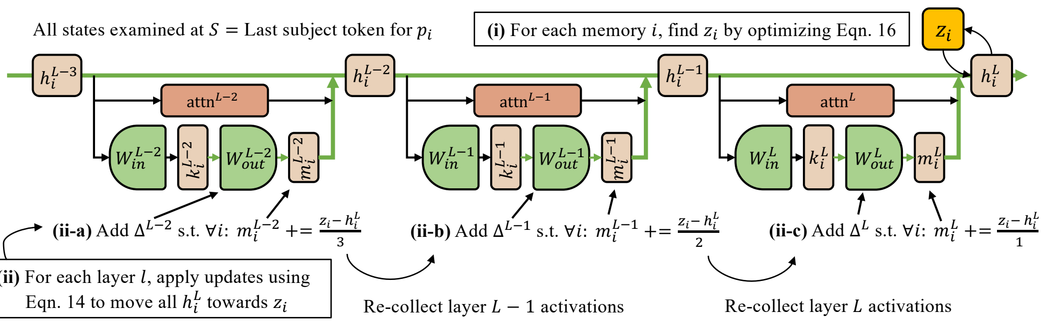
All states examined at S = Last subject token for p_i
(i) For each memory i, find z_i by optimizing Eqn. 16
z_i
h_i^{L-3}
h_i^{L-2}
h_i^{L-1}
h_i^{L}
attn^{L-2}
attn^{L-1}
attn^{L}
W_{in}^{L-2}
k_i^{L-2}
W_{out}^{L-2}
m_i^{L-2}
W_{in}^{L-1}
k_i^{L-1}
W_{out}^{L-1}
m_i^{L-1}
W_{in}^{L}
k_i^{L}
W_{out}^{L}
m_i^{L}
(ii-a) Add Δ^{L-2} s.t. $\forall i$: $m_i^{L-2} \mathrel{+}= \frac{z_i - h_i^L}{3}$
(ii-b) Add Δ^{L-1} s.t. $\forall i$: $m_i^{L-1} \mathrel{+}= \frac{z_i - h_i^L}{2}$
(ii-c) Add Δ^{L} s.t. $\forall i$: $m_i^{L} \mathrel{+}= \frac{z_i - h_i^L}{1}$
(ii) For each layer l, apply updates using Eqn. 14 to move all h_i^L towards z_i
Re-collect layer $L-1$ activations
Re-collect layer L activations

Figure 4: **The MEMIT update**. We first (i) replace h_i^l with the vector z_i and optimize Eqn. 16 so that it conveys the new memory. Then, after all z_i are calculated we (ii) iteratively insert a fraction of the residuals for all z_i over the range of critical MLP modules, executing each layer's update by applying Eqn. 14. Because changing one layer will affect activations of downstream modules, we recollect activations after each iteration.

a set of additional associations. Unlike Meng et al. (2022), we cannot solve our problem with a constraint that adds only a single new association, so we define an expanded objective:

$$W_1 \triangleq \underset{\hat{W}}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \left\| \hat{W} k_i - m_i \right\|^2 + \sum_{i=n+1}^{n+u} \left\| \hat{W} k_i - m_i \right\|^2 \right). \tag{9}$$

We can solve Eqn. 9 by again applying the normal equation, now written in block form:

$$W_1 \begin{bmatrix} K_0 & K_1 \end{bmatrix} \begin{bmatrix} K_0 & K_1 \end{bmatrix}^T = \begin{bmatrix} M_0 & M_1 \end{bmatrix} \begin{bmatrix} K_0 & K_1 \end{bmatrix}^T \tag{10}$$

$$\text{which expands to:} \quad (W_0 + \Delta)(K_0 K_0^T + K_1 K_1^T) = M_0 K_0^T + M_1 K_1^T \tag{11}$$

$$W_0 K_0 K_0^T + W_0 K_1 K_1^T + \Delta K_0 K_0^T + \Delta K_1 K_1^T = M_0 K_0^T + M_1 K_1^T \tag{12}$$

$$\text{subtracting Eqn. 8 from Eqn. 12}: \quad \Delta(K_0 K_0^T + K_1 K_1^T) = M_1 K_1^T - W_0 K_1 K_1^T. \tag{13}$$

A succinct solution can be written by defining two additional quantities: $C_0 \triangleq K_0 K_0^T$, a constant proportional to the uncentered covariance of the pre-existing keys, and $R \triangleq M_1 - W_0 K_1$, the residual error of the new associations when evaluated on old weights W_0. Then Eqn. 13 can be simplified as:

$$\Delta = R K_1^T (C_0 + K_1 K_1^T)^{-1}. \tag{14}$$

Since pretraining is opaque, we do not have access to K_0 or M_0. Fortunately, computing Eqn. 14 only requires an aggregate statistic C_0 over the previously stored keys. We assume that the set of previously memorized keys can be modeled as a random sample of inputs, so that we can compute

$$C_0 = \lambda \cdot \mathbb{E}_k \left[k k^T \right] \tag{15}$$

by estimating $\mathbb{E}_k \left[k k^T \right]$, an uncentered covariance statistic collected using an empirical sample of vector inputs to the layer. We must also select λ, a hyperparameter that balances the weighting of new v.s. old associations; a typical value is $\lambda = 1.5 \times 10^4$.

4.3 Updating multiple layers

We now define the overall update algorithm (Figure 4). Inspired by the observation that robustness is improved when parameter change magnitudes are minimized (Zhu et al., 2020), we spread updates evenly over the range of mediating layers $\mathcal{R}$. We define a target layer $L \triangleq \max(\mathcal{R})$ at the end of the mediating layers, at which the new memories should be fully represented. Then, for each edit $(s_i, r_i, o_i) \in \mathcal{E}$, we (i) compute a hidden vector z_i to replace h_i^L such that adding $\delta_i \triangleq z_i - h_i^L$ to the hidden state at layer L and token T will completely convey the new memory. Finally, one layer at a time, we (ii) modify the MLP at layer l, so that it contributes an approximately-equal portion of the change δ_i for each memory i.

(i) Computing z_i. For the ith memory, we first compute a vector z_i that would encode the association (s_i, r_i, o_i) if it were to replace h_i^L at layer L at token S. We find $z_i = h_i^L + \delta_i$ by optimizing the residual vector δ_i using gradient descent:

$$z_i = h_i^L + \underset{\delta_i}{\operatorname{argmin}} \frac{1}{P} \sum_{j=1}^{P} - \log \mathbb{P}_{G(h_i^L \mathrel{+}= \delta_i)} \left[o_i \mid x_j \oplus p(s_i, r_i) \right]. \tag{16}$$