Enforcing Semantic Constraints in SynthSAEBench Hierarchies

The Approach: Compositional Feature Directions with Hierarchical Constraints

While SynthSAEBench's current hierarchy mechanism only enforces probabilistic dependencies (children fire when parents fire), we can introduce partial semantic structure by modifying how feature direction vectors are constructed within the hierarchy. The key insight is to make child feature directions **compositionally dependent** on their parent directions, rather than having them be independent random vectors. Specifically, when constructing a hierarchical feature tree, we can define each child feature direction as $d_{\text{child}} = \alpha \cdot d_{\text{parent}} + \beta \cdot d_{\perp}$, where d_{parent} is the parent's direction vector, $d_{\perp}$ is a component orthogonal to the parent (representing the "specialization" of the child concept), and α, β are mixing coefficients that control how much of the parent's representation is inherited. This creates genuine compositional structure where activations containing child features literally contain components of parent features in their vector representations. By carefully choosing the α values across the hierarchy (e.g., $\alpha = 0.7$ for closely related concepts, $\alpha = 0.3$ for more distantly related ones), we can encode semantic relatedness as geometric similarity in the feature space.

Implementing d_ :

To get a vector orthogonal to the parent direction, you use **Gram-Schmidt orthogonalization**. You start with a random vector v, then subtract off its projection onto the parent direction:

$d_{\perp} = v - (v \cdot d_{\textbf{parent}}) d_{\textbf{parent}}$

The term (v · d_parent) d_parent is the component of v that "points along" the parent, so subtracting it leaves only the part that's perpendicular. You then normalize the result to unit length. This gives you a vector that lives in the subspace of the full embedding space that is completely "unrelated" to the parent direction — it captures whatever is unique or distinguishing about the child concept beyond what it inherits from the parent.

What does geometrically:

controls how much of that orthogonal "specialization" component makes it into the final child direction. The full formula d_child = · d_parent + · d_ is a **linear combination** of two orthogonal vectors, which means you're essentially picking a point on a 2D plane spanned by those two directions. Since d_parent and d_ are orthogonal unit vectors, the cosine similarity between d_child (after normalization) and d_parent is determined by the ratio / — a large relative to means the child direction tilts strongly away from the parent toward its own unique subspace, encoding a concept that is more "specialized" and less like its parent. A small means the child hugs closely to the parent direction, encoding

a concept that is almost synonymous with the parent. So and together act as a geometric dial controlling where on the spectrum between "identical to parent" and "completely independent from parent" the child concept sits.

Hierarchical Feature Geometry Containing Misalignment Concept Semantics

Step 1: LLM generates the misalignment concept hierarchy. You prompt an LLM to produce a tree of misalignment-related concepts — something like "Deceptive Reasoning" as a root, with children "Goal Misrepresentation" and "Information Withholding," and then grandchildren like "Reward Hacking" and "Sycophantic Agreement" under Goal Misrepresentation, and "Selective Omission" and "Framing Manipulation" under Information Withholding. Critically, you also ask the LLM to assign an value to each parent-child edge, encoding its judgment of *how semantically similar* the child is to the parent — "Reward Hacking" might get =0.4 from "Goal Misrepresentation" since it's a fairly specific instantiation, while "Goal Misrepresentation" might get =0.7 from "Deceptive Reasoning" since it's almost a direct sub-case.

Step 2: Translate the tree into feature directions. You start at the root and assign it a random unit vector d_root. For each child, you compute d_child = · d_parent + · d_ , where d_ is obtained by Gram-Schmidt against the parent, and is chosen such that after normalization the cosine similarity between parent and child matches the the LLM specified (so is essentially derived from , not independently set). You recurse down the tree level by level, so grandchildren inherit geometric structure from both their parent and transitively from their grandparent — "Reward Hacking" ends up with some directional overlap with both "Goal Misrepresentation" *and* "Deceptive Reasoning," which is exactly the right semantic property.

Step 3: Feed into SynthSAEBench's hierarchy mechanism. The resulting feature directions slot directly into the feature dictionary D, and the tree structure maps directly onto SynthSAEBench's existing hierarchy: the parent firing-probability constraint (c_child ← c_child · 1[c_parent > 0]) enforces that "Reward Hacking" can only be active when "Goal Misrepresentation" is active, while the geometric construction ensures that the hidden activations produced for "Reward Hacking" samples literally contain a component pointing in the direction of "Deceptive Reasoning."

Step 4 and additional experiments: You can now train an SAE on this synthetic data and ask very concrete diagnostic questions: does the SAE learn a latent whose decoder direction has high cosine similarity with d_root even when only grandchild features are firing? Does ablating the latent most aligned with "Deceptive Reasoning" impair reconstruction of "Reward Hacking" more than it impairs reconstruction of unrelated features? Does the SAE split the hierarchy correctly or does it absorb child concepts into parent latents? Because you hold the ground truth — you know exactly which direction corresponds to

which concept and what the values were — you can measure failure modes with precision that is impossible on a real LLM. The LLM-generated hierarchy is what makes the concepts interpretable to humans; the compositional direction construction is what makes the geometry testable.

Theory: Semantic Correlation Through Geometric Constraints

The theoretical foundation rests on the principle that **semantic relatedness should manifest as geometric structure in representation space**. When we set $\alpha > 0$, we create non-zero cosine similarity between parent and child feature directions: $\cos(\theta) = d_{\text{child}}^T d_{\text{parent}} = \alpha$ - after normalization, from

$$d_{\text{child}}^T d_{\text{parent}} = (\alpha \cdot d_{\text{parent}} + \beta \cdot d_\perp)^T d_{\text{parent}} = \alpha \underbrace{(d_{\text{parent}}^T d_{\text{parent}})}_{=1} + \beta \underbrace{(d * \perp^T d_{\text{parent}})}_{=0} = \alpha),$$

while their orthogonal components $d_\perp$ point in different directions to distinguish them from each other. This creates a testable prediction: SAEs that successfully decompose these features should discover latents where the decoder directions for child features have high cosine similarity with the decoder direction for the parent feature, and interventions that ablate the parent feature should impair reconstruction of child features more severely than unrelated features.

Limitations and What This Achieves

This approach provides **weak semantic structure**—it's geometrically grounded but still falls short of true semantic understanding. We're encoding human-chosen semantic relationships (like "deception contains goal misrepresentation") into the statistical properties of the data, but the model still doesn't "understand" deception in any functional sense. What we gain is the ability to test whether SAEs can discover and respect compositional structure: if an SAE trained on hierarchically-constrained features with compositional directions fails to learn latents that preserve the parent-child geometric relationships, this tells us it will struggle even more with the implicit semantic hierarchies in real LLMs. Critically, this approach lets us validate necessary conditions for handling semantic structure—if SAEs can't decompose explicitly encoded compositional hierarchies where we control the mixing coefficients and geometric relationships, they certainly won't succeed on the far more complex implicit semantic structures in language model representations. This bridges the gap between purely statistical hierarchies (current SynthSAEBench) and truly semantic hierarchies (which we cannot fully create without solving the interpretability problem we're trying to investigate), providing a testbed for architectural improvements that must handle compositional feature structure.