

What Does Belief State Geometry Look Like in Non-Ergodic Mess3 Components?

Elena Ajayi

elenaajayi@outlook.com

Abstract

Prior work has shown that transformers trained on a single ergodic Mess3 process represent their belief state geometry in their residual stream [4]. However, what happens when real-world data like natural language happens to be non-ergodic, where sequences come from different underlying processes? Current literature has minimal evidence of this – testing whether belief geometry extends to non-ergodic processes where the model is tasked with identifying which specific process generated a sequence, and how beliefs are tracked within that process. To investigate this, I trained a small transformer on a non-ergodic mixture of three Mess3 components with different alpha values. To arrive at my predictions, I looked at the current gaps in the literature and made five pre-registered predictions about the geometry before seeing the results. One of the key results is that the model learns belief states with high fidelity ($R^2 = 0.85$) in a shared subspace across components with an overlap of 0.96–0.99 (silhouette ≈ -0.03) without separating the components geometrically, despite component classification being near chance (53% vs 33%). Additionally, further analysis shows the model slightly beats the oracle (-0.039), outperforming the single component Bayes-optimal baseline. Overall, two of my five predictions were confirmed, one was partially confirmed, and two were refuted.

Keywords belief state geometry · mechanistic interpretability · non-ergodic mixtures · hidden Markov models

1 Why Non-Ergodic Structure Matters

Prior literature shows that transformers trained on a single ergodic Mess3 process represent belief state geometry in the residual stream [4], however, this was only tested on one process at a time, and non-ergodic was left as future work. Real-world data like languages in documents, topics, and authors are non-ergodic, as they generate different processes. Riechers et al. [1] also formalized this phenomenon: the model will see the sequences from a mixture of different ergodic components, and the model will then have to determine which component it’s in. Therefore, this creates two jobs for the model: identify which component generated the sequence, and then track the belief state geometry within that component. This first job could be seen as in-context learning. This eventually matters in the long run, because the way a model represents both tasks in its activation space is indicative of how transformers handle multi-source data.

2 Setup

To set up this transformer, I defined a few sets of parameters: I used three Mess3 components with different alphas = 0.4, 0.7, 0.9, which shared the same $x = 0.1$. Each sequence is then generated

entirely by one component, and sampled uniformly. The model’s sequence length is 16, and it never sees which component generated the sequence. The model’s transformer has three layers, 2 attention heads, with a $d_{\text{model}} = 64$, and 152K parameters. The model is also trained for 15K steps with an Adam optimizer and a learning rate $\text{lr} = 1\text{e-}3$, with a cosine schedule. Additionally, the final loss was 1.031, compared to a uniform entropy of $\ln(3) = 1.099$.

3 Pre-Registered Predictions

The following predictions were written before seeing any experimental results.

To arrive at my predictions, I read through four papers: [4], [2], [3], and [1]. None of them tested a non-ergodic mixture of Mess3 components where each component comes from another component. So my predictions were focused on connecting the findings from each paper and resolving the gaps between them: [3] found orthogonal subspaces but only for simultaneous factors, [2] studied attention and MLPs but only for a single component, and [1] described how models identify components but not the underlying geometry.

Based on the formulas in [1] and [4], the model needs to figure out two things. First, which component generated this sequence. Following [1], this is:

$$P(C = c \mid x_{1:t}) = \frac{Q(C = c) Q^{(c)}(x_{1:t})}{\sum_{c'} Q(C = c') Q^{(c')}(x_{1:t})} \quad (1)$$

As the model sees more tokens, this distribution becomes more concentrated toward the true component. Second, within whichever component the model thinks it’s in, it tracks the belief state. Following [4], this update is:

$$\eta' = \frac{\eta T_c^{(x)}}{\eta T_c^{(x)} \mathbf{1}} \quad (2)$$

where $T_c^{(x)}$ is the transition matrix for token x under component c . Different components have different transition matrices because they have different alpha values, so the belief paths look different for each component. When doing PCA, I would expect to see the fractal patterns overlap from all three components when they occupy the same region in the activation space, rather than three separate clusters.

1. **The belief state representations share a subspace across the layer’s components.** The model will represent all components in the same subspace that are not in separate orthogonal ones, and use the offsets to tell them apart. In [3], the authors found orthogonal subspaces for factors that were all active at the same time. However for my setup, only one component will be active per sequence thus the factors will be exclusive. Since the model will only ever see one component for each sequence, it will never have to hold two components’ belief states at the same time. Hence, giving each component its own subspace would be wasteful. I recommend the model should reuse the same dimensions and then encode the component identity as an offset.
2. **As we get deeper, tracking the model’s beliefs and identifying its components gets easier.** If we use 3 layers for this experiment, the first layer has little information about the component it’s in because it has only seen the raw token embedding. The third layer would have the input of the previous two layers. So even though one layer does both jobs, with attention doing the belief approximation, and the MLP doing the nonlinear correction,

as described in [2], the later layers will produce better component estimates, and clearer within-component fractals because they are working with better inputs.

3. **As the model becomes more context rich, the components will gradually separate in the geometry.** When the model sees more tokens in a sequence, the components will separate gradually in the geometry. The rate at which the geometric separation happens depends on how different the components are. If alphas are far apart like 0.3 vs. 0.7, the model can easily tell them apart in a few tokens, so the separation shows up early. If the alphas are close like 0.5 vs. 0.56, the model will need more tokens to determine which component it is in, so the separation would happen later. This is consistent with [1], which found that when components share the same alphabet and only differ in distributions, their identification is gradual.
4. **Separation of components appear sharply during training when the induction heads form.** The geometry will remain messy until the induction heads form, and then the components sharply separate and become more distinct. This is based on [1], which found that when induction heads form, it causes a phase transition in the model’s training loss.
5. **The final layer is sufficient for recovering the model’s belief geometry.** As described in [4], I would expect that cross-component token collisions could happen, but they are not really built into the structure the way they were in RRXOR. In RRXOR, the multiple belief states are mapped to identical next token predictions. In my setup, this structural property is non-existent, so I expect that collisions would be coincidental. Therefore, the geometry should be mostly recoverable from the final layer without needing to combine activations from all layers into one long vector.

4 Results

Based on the experiments I ran, particularly with respect to the first prediction, the model puts all the three components in the same space. When I measured how much the component subspaces overlap, it was 0.96–0.99, meaning that the model only needed 5 dimensions to capture 96% of what was going on.

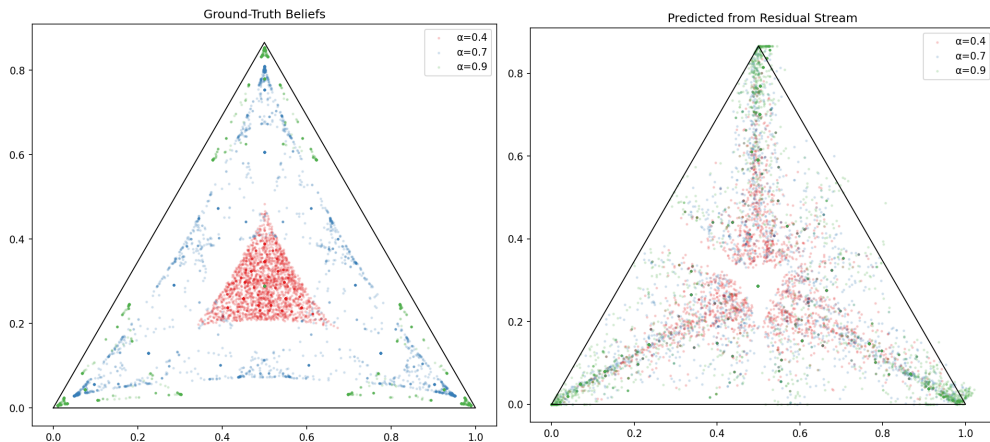


Figure 1: Ground-truth belief states (left) vs beliefs predicted from the residual stream via Ridge regression (right), projected onto the 2-simplex. Colors indicate component identity.

As for the second prediction, the model gets better at tracking beliefs once you go deeper. At the embedding layer, the belief accuracy is 0.618. At the final layer (layer 3), it is 0.842. Therefore later layers are more informative. However, the model does not separate the components at any of the layers, the silhouette score stays negative. Therefore, the components compound, so it does get better with depth but that doesn't mean the components will separate.

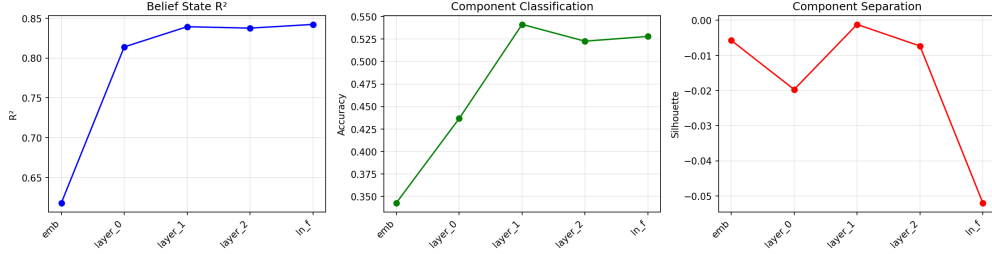


Figure 2: Belief R^2 , component classification accuracy, and silhouette score across layers. R^2 increases while silhouette remains near zero.

For my third prediction, I predicted that the model sees more tokens in a sequence, that would then make the model gradually pull apart in its belief state geometry. However, in the experiments this did not happen; the silhouette score is approximately -0.03 at every context position from start to finish and there was no gradual separation at all. For the fourth prediction, I predicted that there would be a sharp moment during the training where the geometry would sharply separate into clusters when the induction heads form. However this did not happen either; the silhouette score stays around -0.04 throughout the training with no transition phase. The belief state in R^2 just rises smoothly from 0.74 to 0.85 without any sharp transition. For the fifth prediction, I hypothesized that the final layer contains almost everything needed to recover the belief geometry. In the experiments, the final layer R^2 is 0.849, and all the stacked activations from all the layers only give 0.853, the difference is ~ 0.004 , so there might not be a need to look at multiple layers. However, this may change as the layers scale to more layers.

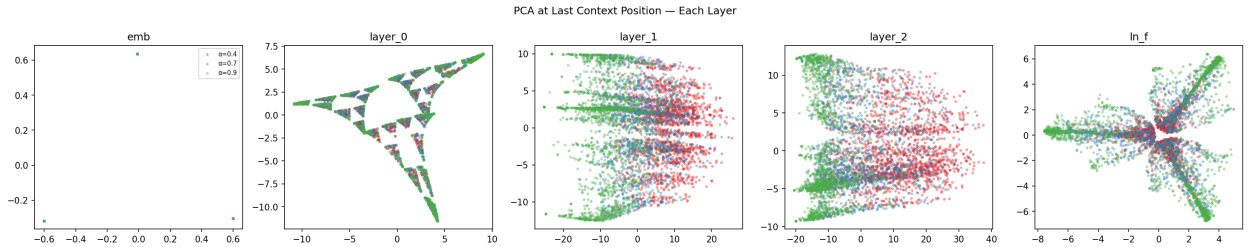


Figure 3: PCA at the last context position for each layer. Components overlap throughout, confirming the shared-subspace geometry.

I believe the third prediction and fourth prediction were wrong because the model shouldn't really need to separate the component geometry because the model's belief state geometries are already different for different alpha values. If the model is tracking the belief states in the $\alpha = 0.9$ component, the path through the model's belief space would look different from the $\alpha = 0.4$ component. The model's component identity is stored in the model's belief state, so separating the model's geometry would be repetitive.

Prediction	Key Result	Confirmed?
1. Shared subspace	Overlap 0.96–0.99	Yes
2. Compounding layers	R^2 rises, silhouette does not	Partial
3. Gradual separation	Silhouette ≈ -0.03 everywhere	No
4. Sharp transition	No jump across training	No
5. Final layer sufficient	R^2 gap = 0.004	Yes

5 Follow-Up Analyses

5.1 Activation Patching

For the additional analysis, I did activation patching: I patched the 3D model’s belief subspace that was identified by the Ridge regression from one component to another. Additionally, the KL divergence between the original and the patched predictions was 0.000, which meant that it had a zero effect. The likely explanation for this is that the model distributes its belief information across more than 3 of the 64 dimensions, so when I attempt to patch a small subspace, this doesn’t really move the predictions.

5.2 MLP vs Attention Decomposition

I also compared MLPs vs attention, and based on my experiments, it was found that MLPs encode more belief information than attention at every layer; MLPs have 0.78–0.82 vs 0.31–0.70 for attention. The attention at R^2 decreases with depth, while the MLP consistently stays high.

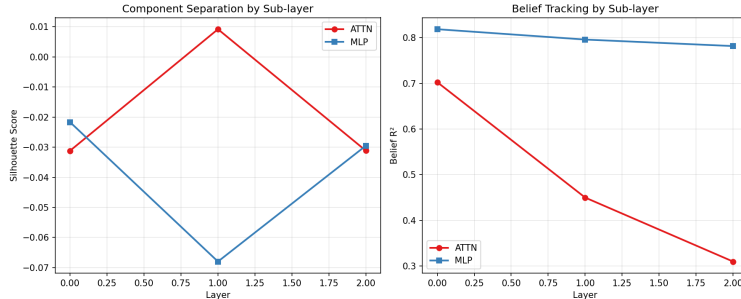


Figure 4: Belief R^2 from MLP vs attention outputs at each layer. MLPs consistently carry more belief information; attention R^2 decreases with depth.

5.3 Bayes-Optimal Baseline

When I investigated the model’s per-position cross-entropy against an oracle that already knows which component generated the sequences, and does the belief tracking, my model’s loss was 1.021, the oracle’s was 1.059, and the uniform guessing was ~ 1.099 . Therefore, my trained model actually beat the oracle by -0.039 , which implies that the strategy of hedging is a better strategy as opposed to committing to one component, especially when the model is unsure.

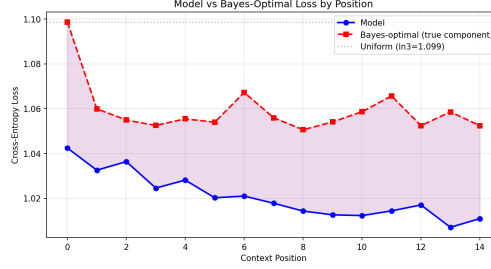


Figure 5: Per-position cross-entropy: model vs Bayes-optimal oracle. The model slightly outperforms the oracle (gap = -0.039) by implicitly averaging over components.

5.4 Cross-Component Transfer Probes

I trained the model’s linear probe to read beliefs from one component’s activations and then tested it on a different component’s activations. Within that same component, I found that the linear probes were quite accurate (0.96), however when extended across components, the probes do not perform well (-3.2 on average), but it was found that this was asymmetric: when comparing with alphas that are closer ($\alpha = 0.7$ and 0.9), the transfer is decent, but anything including $\alpha = 0.4$ collapses (R^2 as low as -14.9). This $\alpha = 0.4$ is the outlier. The geometry is shared based on the results from the experiment that analyzed my first prediction, but the linear probe trained on one component is not able to read beliefs from another component, which means that the model uses different paths of that shared space for different components.

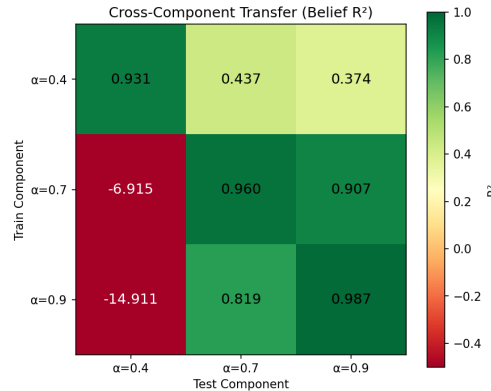


Figure 6: Cross-component transfer probe R^2 matrix. High diagonal ($\bar{R}^2 = 0.96$) but poor off-diagonal transfer, indicating functionally component-specific representations despite shared geometry.

5.5 Out-Of-Distribution Generalization

For further analysis, I generated sequences from a component that was not seen in the training data with $\alpha = 0.6$. I found that when I trained the probe on the original components, it still works on the new one (0.886), and the retrained probe on the OOD data was 0.944, the transfer gap between the probes was 0.059. This shows that the model generalizes well to an alpha it hasn’t seen, and the model learned the general structure of the Bayesian belief updating instead of memorizing.

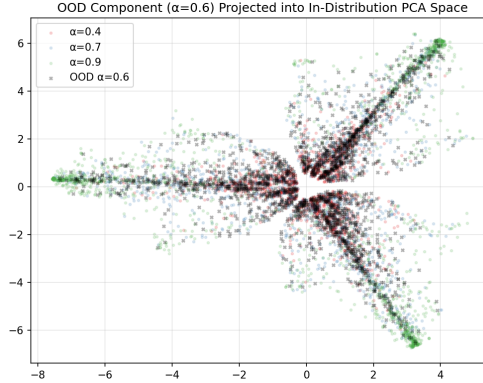


Figure 7: Out-of-distribution generalization to unseen $\alpha = 0.6$. In-distribution probe achieves $R^2 = 0.886$ on OOD data, suggesting the model learned belief update structure.

6 Discussion

The main finding from the subspace analysis and cross-component transfer probes analysis is that the model shares a subspace but maps differently per component. This illustrates that the model doesn't store the identity of its components as its own separate dimension. On the other hand, the identity of its components is encoded into the model path through the shared space. The model also doesn't need to separate the components geometrically because the belief paths are already different for different alpha values. If the paths are already different, then separating them into their own clusters wouldn't be necessary, which also explains why my third and fourth predictions were refuted. My model also beat the Bayes-optimal baseline which suggests that the model found a path that a single component oracle cannot use. Instead of using one component, my model maintains uncertainty over all components and averages its predictions, which is relevant to the phenomena of language models, because language models don't necessarily oversaturate on one topic but generalize across multiple topics. In terms of generalizations, the Out-Of-Distribution generalization results also suggest that the model learned the belief structure itself, rather than simply memorizing three sets of what it was trained on. The linear probe that was trained on the original components works on an alpha that was not seen in the training data, which is evidence that the model learned genuine structure rather than memorizing. The lack of increase in activation patching illustrated that the linear probes can identify subspaces that correlate with belief states, without the subspaces having any causal relationship. This becomes a broader concern for the interpretability field which indicates that results from probing are taken as evidence of how the model works, when in fact they may only reflect what the model encodes.

7 What I Would Do Next

For future work, I will test more components with closer alpha values to find the exact point in the activation space the shared representation diminishes, because at some level in the activation space between the components, the model may not be able to differentiate them through belief geometries alone. Secondly, given the null result from the activation patching, I will run a full residual stream patching as a control. If swapping the full activation changes the model's predictions but swapping the 3D subspace does not, that could confirm that the belief information is spread across more dimensions than the probe identified. Lastly, I will scale these experiments to train larger models (>3) to test whether the separation of components grows as the models become larger. It could be

possible that a larger model that has more dimensions and more layers may geometrically separate the components, even though the small model didn't find it necessary.

References

- [1] Paul Riechers, Sarah Marzen, Adam Shai, and James P. Crutchfield. Next-token pretraining implies in-context learning: A study using epsilon machines. *arXiv preprint*, 2025.
- [2] Adam Shai et al. Constrained belief updates explain geometric structures in transformers trained on hmms. *arXiv preprint*, 2025.
- [3] Adam Shai et al. Transformers learn factored representations along complementary subspaces. *arXiv preprint*, 2025.
- [4] Adam Shai, Sarah Marzen, Paul Riechers, and James P. Crutchfield. Transformers represent belief state geometry in their residual stream. *arXiv preprint arXiv:2405.15943*, 2024.