An Investigation into Energy Minimization Properties of MLP Features in LLMs

Jackson Kaunismaa

University of Toronto

Supervisor: Professor Vardan Papyan

April 14, 2024

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

Understanding the geometry of features in the activations of neural networks remains one of the primary goals of mechanistic interpretability research. In this work, we develop and investigate a preliminary method for analyzing the geometry of features and make several key findings. We show that surprising configurations of features like simplexes arise in the early layers of LLMs. We additionally find that low-dimensional, abstract representations of features can capture the geometric structure of actual features inside Transformers and that these representations are in energy minimizing configurations. We demonstrate that cosine similarity is preferred over RBF similarity in comparing two geometries.

1 Introduction

Over the past few years, Transformer-based large language models (LLMs) have made massive leaps in their capacity for text generation and general problem solving [6, 18, 23, 24]. However, these LLMs remain inscrutable. Attempts to mechanistically understand neural networks in the past have involved determining how and to what extent they are able to represent human-interpretable features in their internal activations [11, 19, 22].

Typically, features are assumed to correspond to some unique direction in activation space. This assumption is reasonable since a large portion of the computation in neural networks is linear, and it has found empirical evidence across a wide range of domains and models [12, 16, 17, 19, 25].

One potential set of feature directions that is natural to choose is the set of basis vectors that are aligned with individual neurons. In a hidden space of dimensionality n, this would give the network the capacity to represent n mutually orthogonal features, each uniquely associated with one of the n neurons. This view, known as the "basis-aligned" case, when applied to search for neurons that are associated with individual features, has been largely successful.

However, this picture is challenged by the existence of so-called "polysemantic neurons." These are neurons that activate across a broad range of seemingly unrelated stimuli, indicating that it is highly unlikely the neuron is being used to represent a single, human-interpretable feature [2].

One hypothesis for polysemanticity is known as the "superposition hypothesis," which states that models may try to represent more features than they have neurons in that layer [9]. If more features are better for accomplishing its task, the network would be incentivized to learn these extra features, at the potential cost of some features "interfering" with each other, so that feature 1 might look a little bit like features 2 and 3 to

the network. Interference is not a problem in basis-aligned networks, where all features are mutually orthogonal and thus can always be distinguished from one another.

[9] go on explain how exactly these features would be arranged in activation space to minimize interference while maximizing the number of features represented. In particular, they claim that in certain ideal conditions, they would be the local minima of energy functions.

While the hypothesis is appealing mathematically, and toy models exhibiting superposition have been developed, there is little evidence of the phenomenon occurring in modern LLMs trained on real world data. Evidence for superposition in modern LLMs and an understanding of the geometry of features in general would represent a huge step towards making these models more understandable, trustworthy, and auditable and would be a major leap forward for mechanistic interpretability.

This work attempts to uncover the geometry of features in the MLP layers of LLMs and in so doing, find evidence of superposition by checking if they are in energy minimizing configurations. We develop a general method for analyzing feature geometries and constructing low-dimensional, abstract representations of features that share the geometry of the original target feature set. We show that not only do these abstract representations accurately capture the geometry of actual features, but that indeed they are in narrow local minima of energy functions.

2 Background

2.1 Toy Models of Superposition

[9] introduces the concept of superposition, explaining how in certain conditions, neural networks may be incentivized to learn more features than they have neurons. In other words, some networks might be trying to embed more features in their hidden layers than they ought to have the capacity (dimensionality) for. They envision features as essentially binary indicator variables: if that human interpretable property of the input is present, the feature will be active, otherwise it will not. As will be seen in subsection 2.3, this may not be the full picture, but binary features will be assumed for the remainder of this work.

The authors introduce the concept of **feature importance**, which for our purposes represents how "useful" a given feature is towards whatever task the neural network is being trained on.

The choice to represent a feature for a model depends on how much representing

that feature would decrease its expected loss. This is a trade off between **interference**, which intuitively represents how much features get confused for one another by being non-orthogonal, and feature importance.

It can be shown that finding the optimal arrangement of class means in the last layer is equivalent to the minimization of an energy function defined in that activation space. There is some work exploring what particular energy functions might be reasonable in cases similar to superposition, where the network is looking to minimize interference between features [14, 15].

These energy minimization problems, known as Thomson problems, generally have solutions with striking geometries, like the uniform polytopes discussed in [9]. For example, they show that if their toy network tries to represent 4 features using 3 dimensions, it will result in a tetrahedron, or if tries to represent 5 features in 2 dimensions, it will result in a pentagon. These sorts of geometries are the exact ones we are looking to find in our LLMs. Finding such geometries would be strong evidence for the superposition hypothesis, and may even reveal important structure in how LLMs understand natural language.

One consideration when representing features in superposition is the correlation structure of those features. If features are positively correlated, and the model has capacity (dimensionality) to spare, it prefers to make them orthogonal to one another. If it does not have capacity (dimensionality) to spare, it will tend to put them next to each other or "collapse" the set of correlated features into a single representative feature. If features are anti-correlated, the model prefers to make them antipodal to one another. The motif of antipodes appears elsewhere in the literature, and appears to be an extremely common way to put two anti-correlated features in superposition [9, 12, 19]. Since features in the real world almost certainly are not independent from one another, consideration of these principles is important when searching for the geometries predicted by superposition in real models.

Finally, non-uniform feature importance, which is almost certainly the case for real world features, is also an important consideration, because it may lead to deformation of the geometry from the idealized uniform importance case, or may result in a different geometry altogether, with very important features being given their own dedicated neuron, to minimize interference.

2.2 Finding Neurons in a Haystack: Case Studies with Sparse Probing

Extracting feature directions from hidden layers is a non-trivial task made more difficult by polysemanticity. One approach for extracting features from the hidden layers of LLMs is outlined by [13]. They build a dataset with several categories of features consisting of sentences taken from the Pile [10]. A detailed description of some of the relevant feature categories are provided in subsection 3.2. They use these sentences and feature labels to build an activation dataset by passing each sentence into the LLM and extracting the activations from the specific tokens that the labels indicate contain or do not contain the feature. Given these activation vectors, they then train linear probes to predict the label that indicates whether the feature was present in that token or not. The weight vectors of the *probe* can then be extracted and for our purposes will correspond to the direction in activation space associated with that feature.

One question is from what parts of the network should activations be extracted to search for these geometries. [9] define the concept of a privileged basis as the case where features are encouraged to be aligned with the neuron basis. Most layers in a Transformer are not in a privileged basis since they are invariant to rotation. However, the element-wise GELU non-linearity of the MLP layers breaks this rotation invariance and induces a privileged basis. Feature directions are then likely to consist of linear combinations of a *small* number of neurons, where the selection of neurons tells you what particular subspace the network has chosen to represent that feature in. —namely, that subspace whose basis consists of the one-hot vectors associated with those neurons. Training k-sparse probes on the activations will give this neuron subset as well as the feature direction within the subspace defined by those neurons, and so [13] focus mostly on this class of probes.

They find evidence of superposition in early Transformer layers on the "compound word dataset" in the Pythia-70M model [3]. They identify three neurons in the early layers of Pythia-70M that individually are not associated with the bigram "social security", but, when their activations are summed, are strongly associated with it. This indicates that a linear combination of a small number of neurons is being used to represent the "feature" that indicates the presence/absence of the bigram "social security," which is exactly what would be expected if that feature were in superposition. However, [13] do not explore the geometry of their extracted features, and focus instead on the performance of their probes in terms of accurately predicting model behaviour.

2.3 Towards Monosemanticity: Decomposing Language Models With Dictionary Learning

Another approach for extracting features from the hidden layers of Transformers is explored by [5]. They train a 1-layer sparse autoencoder (SAE) on the post-GELU activations of a 1-layer Transformer. By using an embedding size m that is larger than the dimensionality of the MLP n, combined with the sparsity penalty, they encourage the SAE to learn a sparse decomposition of the activation space of the MLP. If the MLP is really using human-interpretable features in its hidden layers, then the simplest decomposition to learn is that which extracts those features and aligns them with individual neurons in the SAE embedding layer.

In practice, this amounts to taking features out of superposition and having them aligned with the basis neurons of the SAE. Their approach, combined with an automated interpretability method that labels SAE neurons based on their highly activating samples [4], allows them to extract features that are seemingly responsible for a very large portion of that MLP's behaviour. Importantly, they are able to do this *without* a labelled dataset of text. Instead, massive, Internet-scale, unlabelled text datasets like the Pile [10] can be used in their entirety. This allows for the method to in theory extract all the features the MLP is using, simply by scaling up the size of the SAE embedding layer m, rather than relying on a small selection of hand-picked features as in [13].

This scalability does not come without its drawbacks though: the training procedure for SAEs is extremely compute heavy, even for the simple 1-layer Transformer they study. To ensure their SAEs are trained across a broad range of contexts, they use activation vectors from 8 billion tokens, compared to just 20 million in the dataset used by [13]. While other work [7] indicates that the SAE approach can work on much smaller datasets as well, on the order of 50 million tokens, we focus on the [13] method as a first approach.

The authors of [5] do some analysis of the geometry of the features the extract and find evidence somewhat contrary to the view outlined in subsection 2.1. In particular, they claim that features may instead group into clusters of related features, rather than as distinct vertices of polytopes. This could be related to the phenomenon of correlated features being "collapsed" into a single feature when the network lacks capacity to represent them orthogonally, as described in subsection 2.1. Another explanation is that the picture of features as indicator variables is false. Instead, features could lie on "feature manifolds" where nearby points correspond to slightly different values of the feature. For example, [20] find a family of neurons that define a manifold over curve/edge orientation in the early layers of CNN vision models. If knowing the exact value of the feature on the

manifold becomes important, more and more capacity might be dedicated to it, and in the limit, might result in a "discretization" of the feature manifold where features become indicator variables again and exhibit the same geometry outlined in subsection 2.1.

3 Methods

Our high-level strategy to determine to what extent the **target** features extracted from actual Transformers are in an energy minimizing configuration will be to first create a low-dimensional set of features that share the same geometry as the target features. These low-dimensional features will be referred to as **abstract** features. Then, we will run an additional energy minimization step on the abstract features to see whether they exist in an energy minimizing configuration. The low-dimensional abstract features are necessary to compute since it would be unrealistic to optimize for energy in high-dimensional space, since we would ignoring interference with other features not in the set we are considering. By restricting to low-dimensional space, we implicitly account for this.

3.1 Feature Extraction

Due to the simplicity and low compute requirements of the method outlined by [13] for feature extraction, we focus on this approach for the remainder of the paper. We began by extracting an activation dataset for the feature datasets provided by [13], using their code and scripts. Then, we train logistic regression probes with L2 penalties to predict the presence of one of their features. The weight vectors of the logistic regression probes then become the feature directions we associate with whatever concept they were classifying, i.e. the target features.

3.2 Analysis of Datasets

The sparse probing dataset consists of a subset of the Pile dataset [10] annotated according to several sets of features defined by [13].

The text features dataset contains several basic textual features, as outlined in Table 1. The negative labels are randomly sampled tokens that don't fall into the given class, and the positive tokens are randomly sampled tokens that obey the rule. These features are typically token-level, not requiring any extra context.

Table 2 shows examples and the descriptions for the Wikidata style datasets. These are created by comparing a JSON dump of famous peoples' names and attributes, scraped from Wikipedia, to the categories of features. The names were then searched for in the Pile dataset and undersampled so that no single name appeared more than a few times. The last token of their name is chosen as the positive sample. Negative samples from the last token of names that are not part of the same class. Since these names may be associated with other people, these features are context dependent, and can't be determined

off the basis of a single token.

One important note is that the names chosen are famous people in general, not necessarily people that are famous for being in their particular category. For example, Niels Bohr, a Nobel Prize-winning Danish theoretical physicist and pioneer in the field of quantum theory, is identified as an association football player, despite being far more famous for his other endeavours.

Feature Name	**Example Sentence**
contains_digit	style in following format?\n03.00, **02**.04\n\nif i Set \n
all_digits	not differ from a normal population. From 5 to **30** months, there appeared a significant probability of intellectual
contains_capital	housing in University Suites and University Apartments. **More** recently, Alpha Psi Lambda, a
leading_capital	() is a rural locality (a village) in **Sem**izerye Rural Settlement, Kaduys
all_capitals	LIMITED TO, THE\n* IMPLIED WARRANTIES OF MERCHANTABILITY **AND** FITNESS FOR A PARTICULAR\n* PURPOSE ARE DISCLAIM
contains_whitespace	lines of evidence have supported the idea that capping **protein** blocks the barbed end of actin filaments,
has_leading_space	.56).The maxillary sinus was most commonly **involved**, followed by the **nasal** cavity (51%
no_leading_space_and_loweralpha	, currentForecast = currentPageViewController.fore**cast**,\nlet currentIndex = indexOf
contains_all_whitespace	(input.hasNextLine())\n}\n\n__System.out.println(
is_not_alphanumeric	\n# yargs command completion script\n#\n# Installation**:** {{app_path}} completion
is_not_ascii	az\nKırkkaşık\nK**ı**rmataş\nKoçbaba\n

Table 1: Description and samples for the text features dataset. The tokens corresponding to positive labels are **bolded**.

3.3 Feature Geometry

One way to identify the geometry that exists between a set of features is through pairwise similarity matrices (PSMs). Similarity functions $d_p : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ should, broadly speaking, output small values for dissimilar inputs x, y and large values for similar ones. Given a similarity function, the i, j-th entry of the PSM of a set of features $\mathcal{X}$ can be computed as:

$$d_p(x_i, x_j) \quad \forall x_i, x_j \in \mathcal{X}. \tag{1}$$

Possible Value	Positive Sentence Example
baseball	Playing baseball professionally, sparked by reportedly meeting Babe **Ruth**
association football	Rutherford's colleague, Danish theorist Niels Boh**r**
basketball	Most asked questions of his wedding day party. Was LeBron **James**
ice hockey	Based on the popular comic strip by Charles M. Schul**z**
American football	1976 election, a Washington elector pledged to President Gerald **Ford**

Table 2: Description and samples from the athlete occupation dataset. The tokens associated with positive labels are **bolded**.

Similarity functions are invariant to orthogonal transformations of their inputs, meaning that if all features underwent the same orthogonal transformation, the PSM would be unchanged. Similarity functions are also symmetric, which means that PSMs are always symmetric matrices. We consider two different types of similarity function in this work, as seen in Table 3.

Similarity Function	Formula
Cosine	$\frac{x_i \cdot x_j}{\|\|x_i\|\|_2 \|\|x_j\|\|_2}$
RBF	$\exp\left(\frac{-\|\|x_i - x_j\|\|_2^2}{C}\right)$

Table 3: The two similarity functions considered in this work. x_i and x_j are the two features being compared. For the RBF kernel, C is a constant chosen to give PSMs with higher contrast in their values. Empirically, we find that $C = 200$ gives satisfactory behaviour.

We say that the geometry of two sets of features $\mathcal{X}$ and $\mathcal{Y}$ are the same if the distance between the PSM of $\mathcal{X}$ and $\mathcal{Y}$ is small. One complication with this is that PSMs are not permutation invariant. If we compare the PSM of $\mathcal{X}$ to the PSM of a feature set consisting of the same features as $\mathcal{X}$ but in a different order, we would find them very different. We therefore define the distance between two PSMs as the minimum Euclidean distance over all possible permutations of the PSMs:

$$d_{\text{PSM}}(T, S) = \min_P ||P \circ T, S||_2, \tag{2}$$

where T and S are both PSMs, and $P \circ T$ denotes applying a permutation to T. We will usually divide this distance by N^2, in order to get an "average error per feature pair" that is comparable between feature sets of different sizes. Since searching over all possible permutations P is infeasible, we develop an algorithm to approximately find this permutation in $\mathcal{O}(N^2)$ time Appendix A.

3.4 Energy

The central claim of [9] that we are investigating is that features will be arranged in energy minimizing configurations. To verify this, we need some way of measuring the energy associated with a given configuration of features. We formalize this with an energy function $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$. Energy is high if features interfere with each other, and low if there is minimal interference. We compute the energy of a set of features $\mathcal{X}$ by taking the sum of the energy function evaluated at all pairs of features in $\mathcal{X}$. [9] states that uniform polyhedra are expected only if feature importance is uniform. To model non-uniform importance for our abstract features x_i, we attach an importance weight w_i to each that is applied to the energy function, giving us the following form for the weighted energy of a set of features $\mathcal{X} = \{x_i : i \in [1, 2, \cdots, N]\}$:

$$\mathcal{E}(\mathcal{X}) = \sum_{i,j} K(x_i, x_j) w_i w_j. \tag{3}$$

We consider 3 different types of energy function $K(\cdot, \cdot)$ in this work, as seen in Table 4. Note that the $-\log$ and Riesz-s kernels depend on the norm of the features x_i, whereas the exp kernel does not.

Energy Function	**Formula**
exp	$\exp\left(\frac{x_i \cdot x_j}{\|\|x_i\|\|_2 \|\|x_j\|\|_2}\right)$
$-\log$	$-\log \|\|x_i - x_j\|\|$
Riesz-s	$\text{sgn}(s)\|\|x_i - x_j\|\|^{-s}$

Table 4: The 3 energy functions considered in this work. x_i and x_j are the two features being compared.

3.5 Joint Optimization

In order to determine both the abstract features x_i and the feature importance weights w_i, we construct an objective that jointly optimizes both:

$$\sum_{ij} K(x_i, x_j) \tilde{w}_i \tilde{w}_j + \lambda_d \sum_{ij} (d_p(x_i, x_j) - d_p(y_i, y_j))^2, \tag{4}$$

where y_i are the target features. The second term in Equation 4 represents to what extent the abstract features share the geometry of the target features y_i, and is the squared distance from the PSM of the abstract features to the PSM of the target features. To ensure that this is primarily optimized for, we typically set λ_d to some large value, on the order of 1000. While we optimize over unconstrained feature importance weights w_i, in order to maintain numerical stability, we restrict the weights used in the weighted energy term of Equation 4 to be positive and bounded by taking a softmax: $\tilde{w} = \text{softmax}(w)$. Minimizing this objective is quite sensitive to the optimizer used. We found that the

conjugate gradient method from scikit-learn's minimization suite most consistently found abstract features that matched the target PSM best [21].

3.6 Re-optimization

Once good abstract features have been found, we want to see if they are in energy minimizing configurations. To do so, we would like to optimize the weighted energy on the abstract features, i.e. to minimize the following objective:

$$\sum_{ij} K(x_i, x_j)\tilde{w}_i\tilde{w}_j + \lambda_x \sum_i \text{abs}(||x_i|| - n_i), \tag{5}$$

where n_i are the norms of the abstract features at the end of the joint optimization step. The first term of Equation 5 ensures that the energy of the abstract features is reduced, while the second term ensures that the norms of the abstract features stay about equal to what they were at the end of joint optimization. Since several of our energy functions decrease with increasing norm of features, this is necessary to ensure that abstract features stay bounded in norm. Similar to joint optimization, this step is also quite sensitive to the optimizer used. We found that the Powell method from scikit-learn's minimization suite was able to achieve the lowest energy values [21].

Importantly, the abstract features in this step are initialized to the output of the joint optimization step, and the importance weights are frozen. If we start from a configuration that matches the geometry of the target features, and then optimize for energy alone, we can measure the distance to the target PSM after re-optimizing for the energy. If the distance is large that indicates that the geometry of the target features is not energy minimizing. If it the distance is small, that indicates that the geometry of the target features is energy minimizing.

3.7 Feature Deduplication

One potential failure mode of the procedure described in subsection 3.6 is that when constructing the abstract features, we assume that all target features are mutually exclusive. If, for example, there were were two target features that were identically represented, then they would have very high energy/interference between them, since they would point in the same direction. Since they represent the same underlying concept, this is not a problem for the Transformer, but our re-optimization step would (incorrectly) want to push these features apart from each other. For example, the features associated with contains_capital and leading_capital in the text features dataset are almost exactly the same. To deal with this, we allow a "feature deduplication" step before passing a given target PSM into the joint optimization step. This involves identifying off-diagonal entries of the target PSM that are above some threshold similarity value, and then dropping one

of the features associated with that pair. For our text features dataset, this means dropping the features leading_capital (duplicate of contains_capital), contains_all_whitespace (duplicate of contains_whitespace), and contains_digit (duplicate of all_digits).

4 Results

We focus on the Pythia-1B-deduped LLM only, due to computational constraints. It has 805M parameters across 16 layers, with an MLP hidden size of 8192. We specifically focus on the post-MLP layers as they are considered to be likely candidates for superposition due to the existence of a privileged basis coming from the activation function [9]. We extracted features for both the "text features" dataset (Table 1) and the "athlete occupation" dataset (Table 2). Other feature sets from [13] were either too small to lead to meaningful analysis, initial analysis did not indicate any particular structure, or, as in the case of Wikidata Occupations, exhibited similar geometry to a dataset already represented—i.e. Wikidata Athlete Occupations.

4.1 Actual Feature Geometries

We analyze the geometric structure of the two extracted feature sets by displaying their PSMs for both choices of similarity function (see Table 3) across several layers of the Transformer. Each entry in the heatmap corresponds to the similarity between a pair of two features. The cosine similarity plots show that geometry is typically established very early on in the network and is maintained throughout all layers (see Figure 1 and Figure 3), with small deviations in early and late layers. The norm-dependent similarity function, i.e. the RBF kernel, shows that in contrast, the norms of features are not constant throughout the network. To contrast the actual features with comparable, random vectors in the same high-dimensional space, we also compute "permuted" feature sets, which simply involve applying some individual permutation to each feature vector (see Figure 2 and Figure 4). These will form the "control groups" for our optimization procedure (see subsection 4.3). Ideally, we should find that configurations associated with actual feature sets are energy minimizing, and configurations associated with permuted feature sets are not energy minimizing.

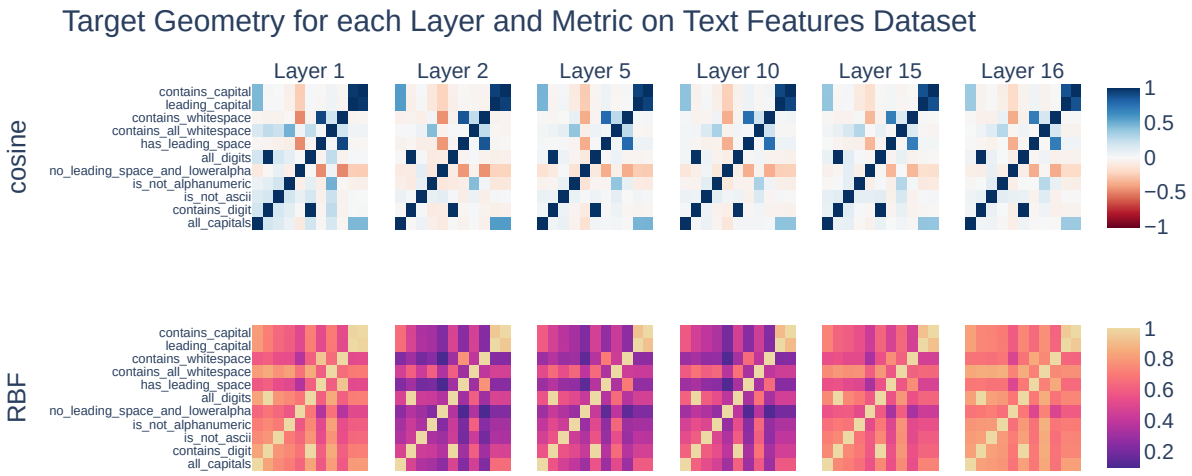

Figure 1: Geometric structure of the text features dataset. We can see that there is non-trivial structure. The x and y axes of each heatmap both correspond to different particular features. For example, the feature pair associated with the dark blue entry in the cosine plots and x=1, y=5 in the middle left is (all_digits, contains_digit)(see Table 1).

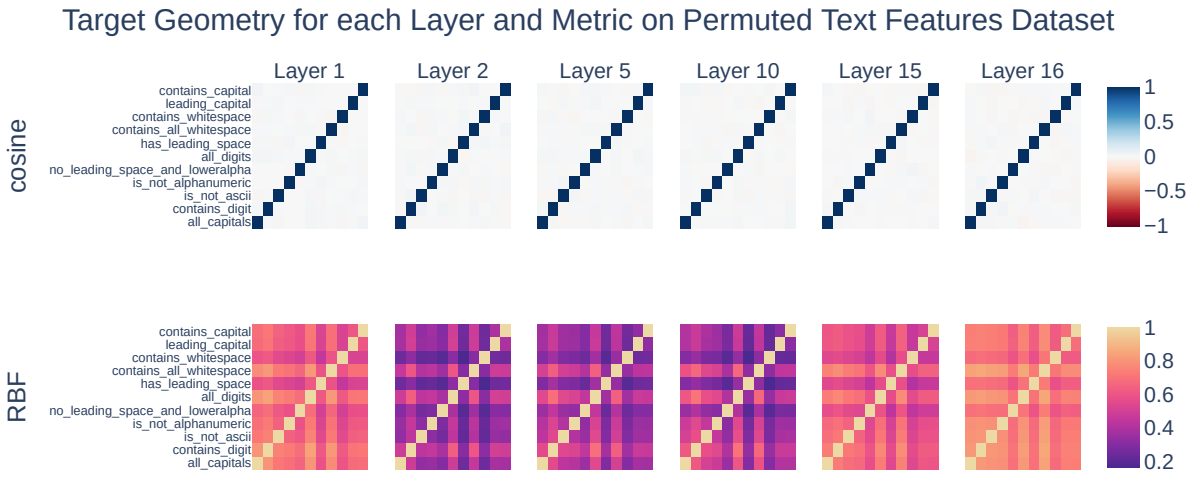

Figure 2: Geometric structure of the permuted features from the text features dataset. As can be seen in the cosine plots, all features are approximately orthogonal and there is little geometric structure, as two random vectors in high-dimensional space are very likely to be orthogonal.

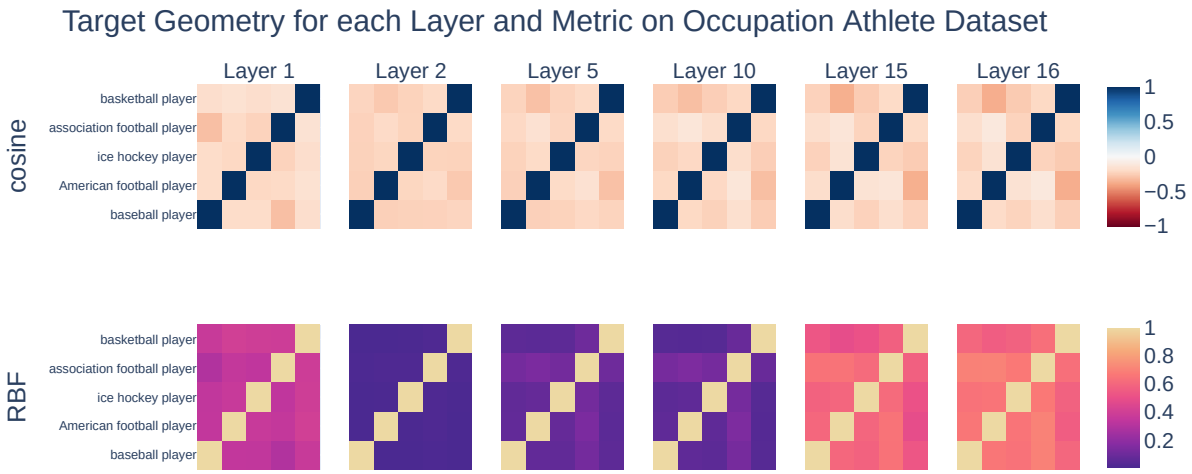

Figure 3: Geometric structure of the athlete occupation dataset. We can see that there is non-trivial structure. Every pair of features has roughly the same, negative dot product, indicating a simplex structure.

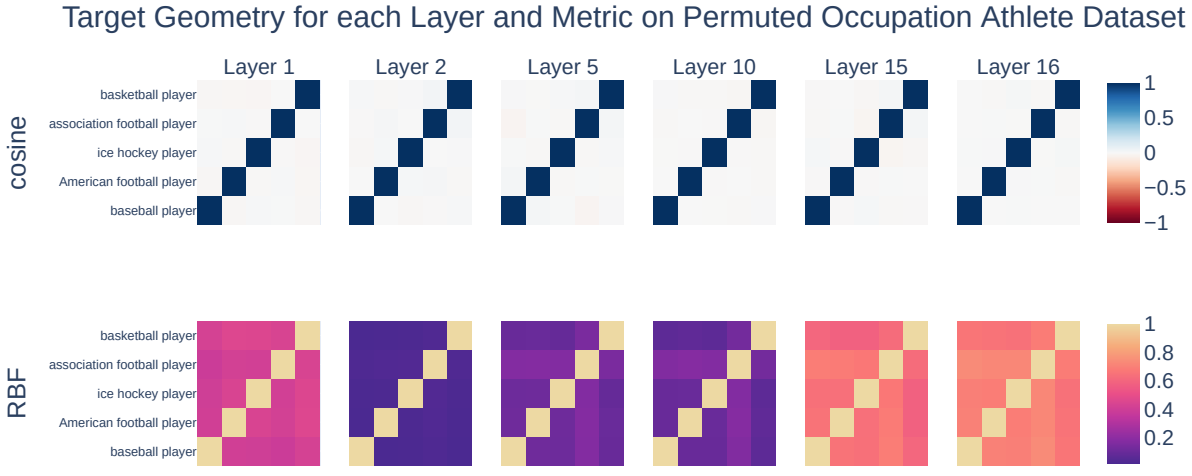

Figure 4: Geometric structure of the permuted features from the athlete occupation dataset. As can be seen in the cosine plots, all features are approximately orthogonal and there is little geometric structure.

4.2 Energy Minimization Experiments

We run hyperparameter optimization over the dimensionality of abstract features m, regularization parameters (λ_d, λ_x), energy functions ($-\log$, exp, Riesz-s), and similarity functions (cosine, RBF) on both the text features and athlete occupation dataset. We also analyze the width of energy minima by applying random perturbations to the initialization in the re-optimization step. Specifically, we add isotropic Gaussian noise to abstract feature x_i with standard deviation σn_i, where σ is the perturbation scale, and n_i is the norm of x_i coming from the joint optimization step. Based on the results from subsection 4.1, which suggest that geometry is about the same across layers, we restrict ourselves to a small subset of the layers (namely layers 8 and 15) to reduce the computation required. As can be seen in Figure 5 through Figure 8, the two step energy minimization procedure largely works. For each setting of (layer, similarity function, energy function), we display the abstract PSM that achieved the lowest distance to the target PSM over all other hyperparameter choices (λ_x, λ_d, and dimensionality of abstract features m).

After the re-optimization step, abstract PSMs still match the target PSM quite well. For larger perturbation scales, the procedure does not work as well, indicating that energy minima may be quite narrow, or that the norm regularization term in Equation 5 overly constrains the abstract features. Generally, the geometry of the athlete occupation features appears closer to an energy minimum than the text features, since the error per feature pair is lower. All of our abstract features are restricted to having strictly fewer dimensions than there are features. If the dimensionality of features were equal to the number of features, then matching the geometry of the target PSM becomes trivial. These plots indicate that lower dimensional representations can accurately capture the geometry of our very high dimensional target features, which is one of the key predictions of [9].

One disappointing result is that the feature importance weights are largely ignored in almost all cases, regardless of hyperparameter settings. This is either a problem with the optimization procedure, or may indicate that our actual features are all of equal importance.

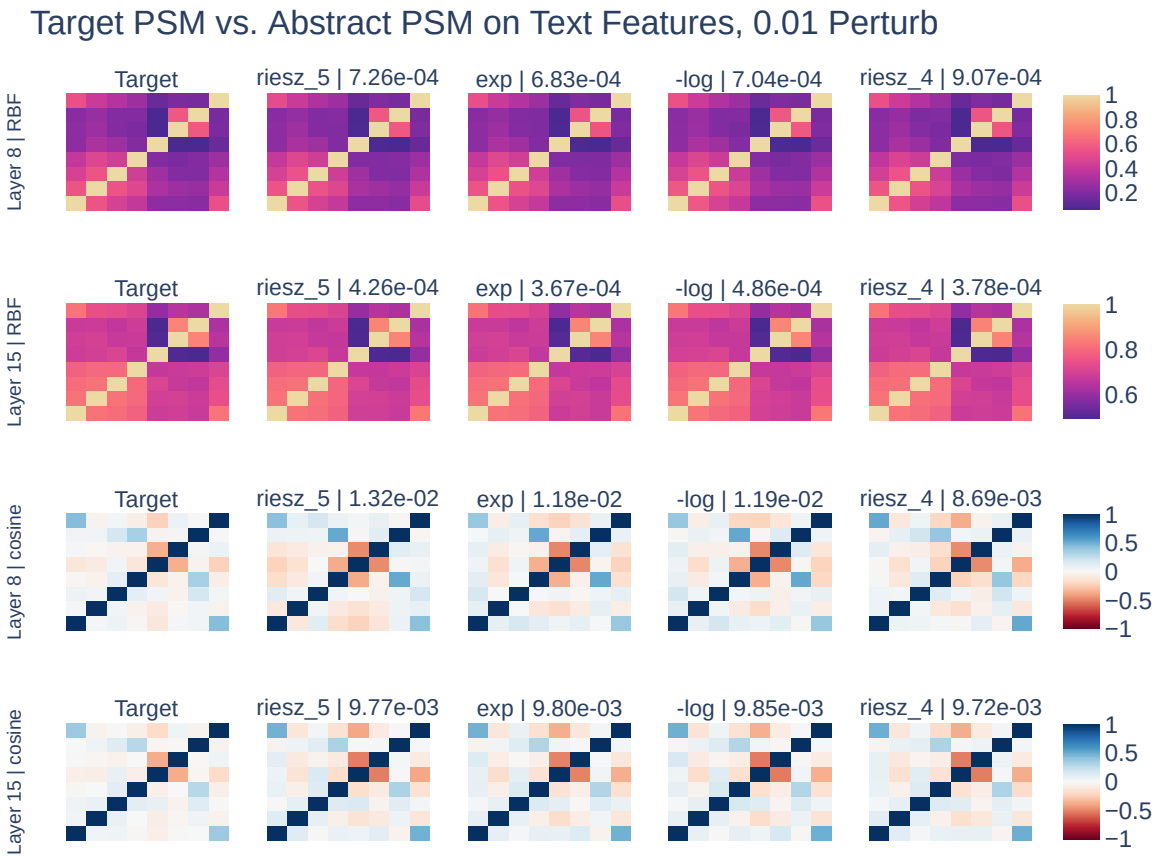

Figure 5: The PSMs of the abstract features after re-optimization compared against the targets for the de-duplicated text features dataset for each energy function, similarity function, and layer, in the perturbation scale 0.01 case. The number in the title of each plot is the Euclidean distance to the target PSM in that row, divided by the number of features squared. This can be thought of as an "error per feature pair". Note that this distance is not comparable between similarity functions since the units are different. We only display the abstract PSMs that achieved the lowest "error per feature pair" for a given choice of layer, similarity function, and energy function. We can see that re-optimization tends to leave the PSM of the abstract features very similar to the target PSM, indicating that the geometries are somewhat close to energy minima.

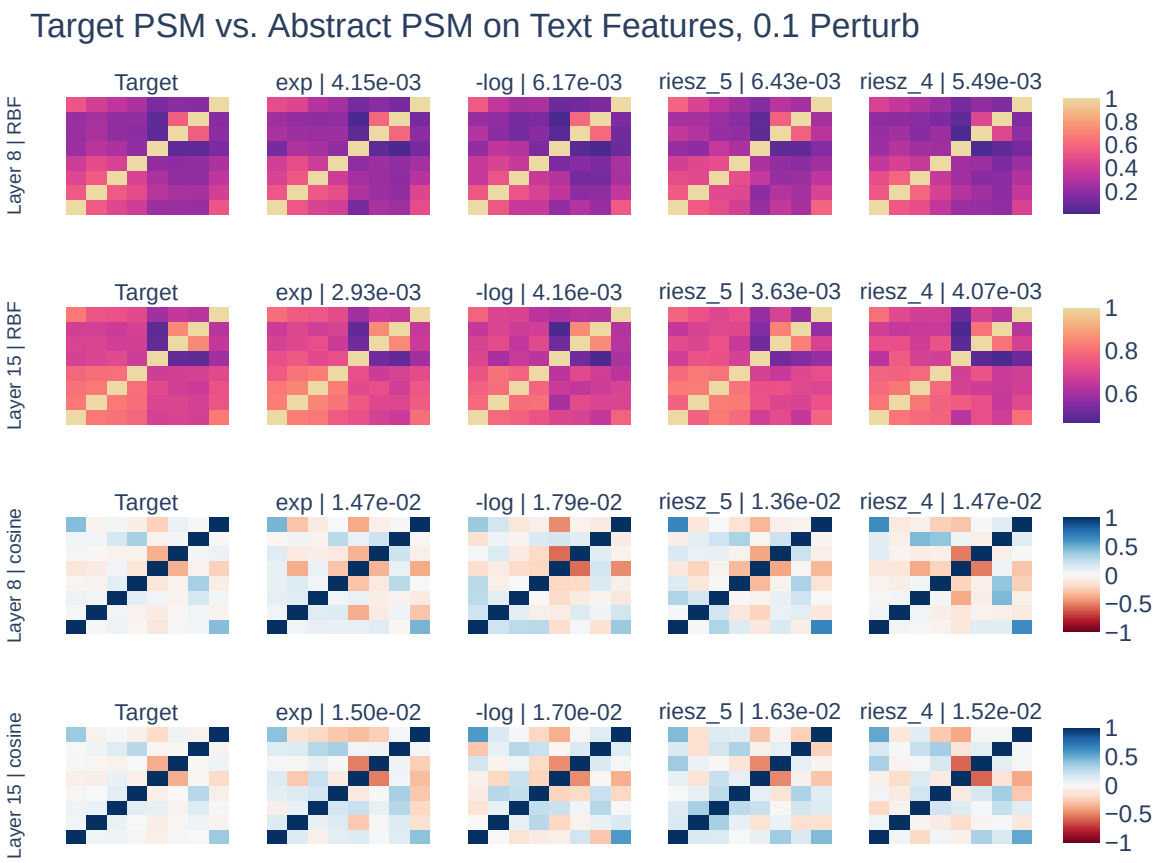

Figure 6: The PSMs of the abstract features after re-optimization against the targets for the de-duplicated text features dataset for each energy function, similarity function, and layer, in the perturbation scale 0.1 case. Increasing the perturbation scale makes the match significantly worse, indicating that these energy minima might be quite narrow.

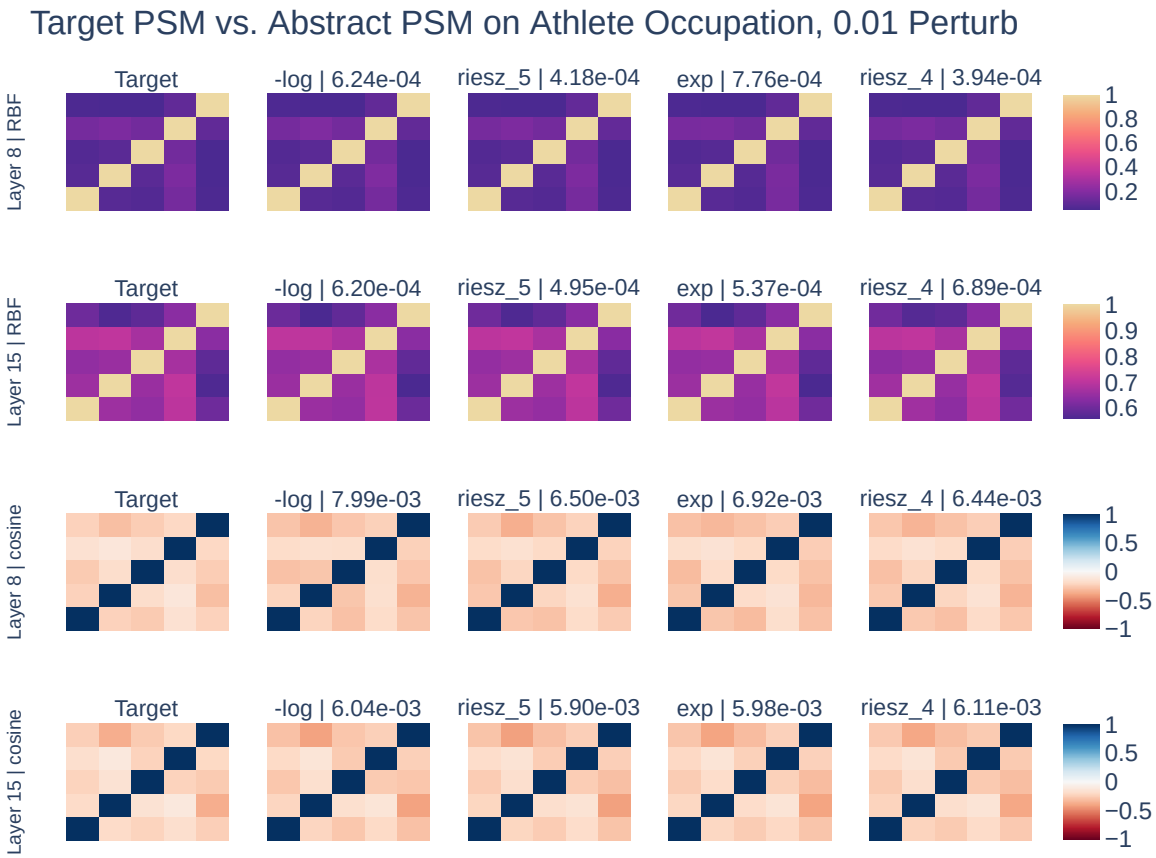

Figure 7: The PSMs of the abstract features after re-optimization against the targets for the athlete occupation dataset for each energy function, similarity function, and layer, in the perturbation scale 0.01 case. The matching is slightly better for athlete occupation as compared to text features, since the error per feature pair is lower. The abstract features are able to match the geometry of the target features even after re-optimization, indicating again that the target features are close to an energy minimizing configuration.

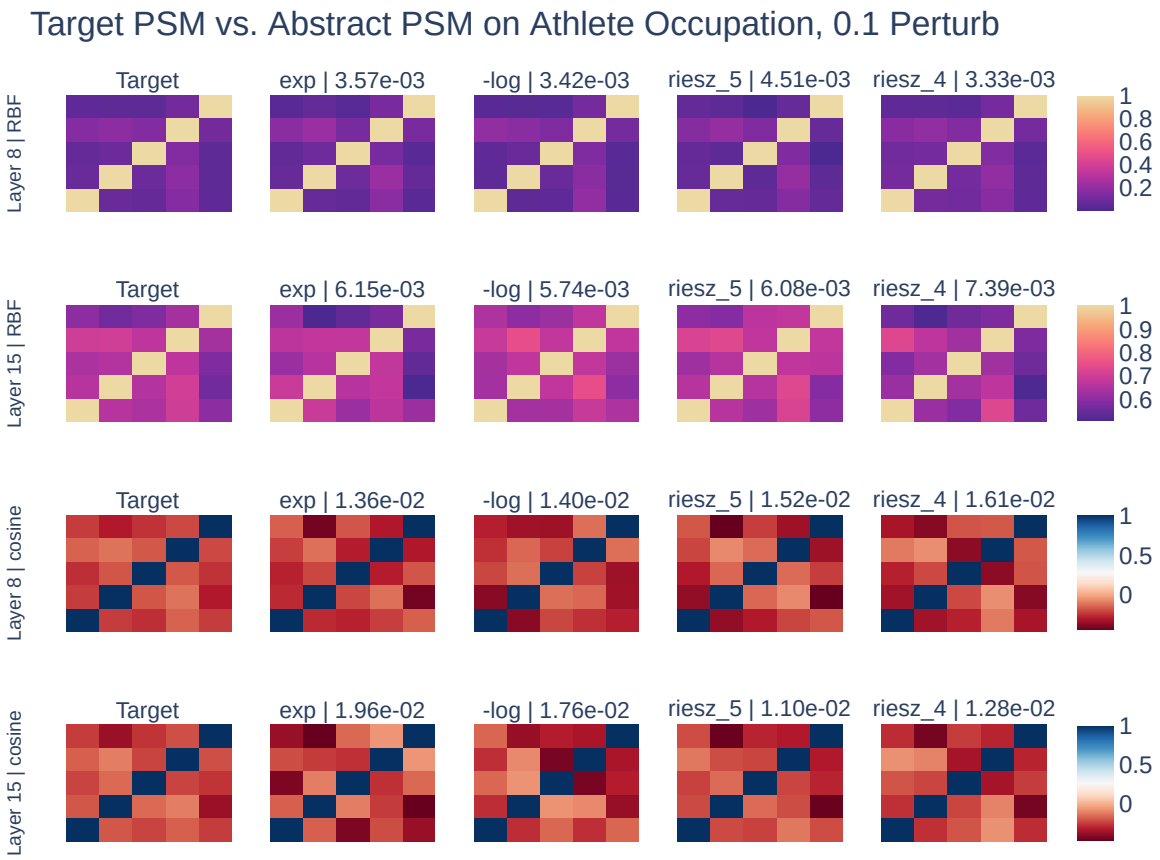

Figure 8: The PSMs of the abstract features after re-optimization against the targets for the athlete occupation dataset for each energy function, similarity function, and layer, in the perturbation scale 0.1 case. Again, the error per feature pair increases significantly when compared to the perturbation scale 0.01 case, indicating that the energy minima are narrow.

4.3 Perturbation Analysis

In order to get a more comprehensive overview of how the perturbation scale affects the distance to the target PSM after re-optimization, we construct distance vs. perturbation plots. For each setting of layer and similarity metric, we pick the optimal hyperparameters (λ_x, λ_d, feature dimensionality, energy function) and vary the perturbation scale. Additionally, we compare the results when the targets come from the permuted and the regular feature sets.

We can see from Figure 9 and Figure 10 that the smaller the perturbation scale, the closer the final abstract features are to the target PSM. The comparison of the permuted features to the actual features demonstrates an important difference between the RBF and cosine similarity functions. For RBF similarity, the final distance as a function of the perturbation scale is approximately the same whether the target features are permuted or not.

At the end of the joint optimization step for permuted features, our abstract features will be in an orthogonal configuration, matching Figure 2. However, we know that orthogonal geometries are not energy minimizers, since they only use a small fraction of available space [9]. Therefore, at the end of the re-optimization step, our features should decidedly not be in an orthogonal configuration. Indeed, for cosine similarity, we can clearly see this in Figure 9 and Figure 10, since the distance to the target PSM is much higher for permuted features than it is for actual features, indicating that energy minimization significantly perturbs the abstract features away from the target PSM in the permuted case. However, for RBF similarity, the distance to the target PSM is about the same, which would indicate that their geometries are about the same. This indicates that cosine is better for distinguishing two geometries, especially when comparing between energy minimizing and non-energy minimizing geometries.

One potential complication to this is the particular way that our "random" features were chosen. Since a permutation would preserve the norm, and RBF depends on the norm of the features, it might be the case that RBF is mostly just a measure of the norm of two features and so would not be a good basis on which to compare geometries.

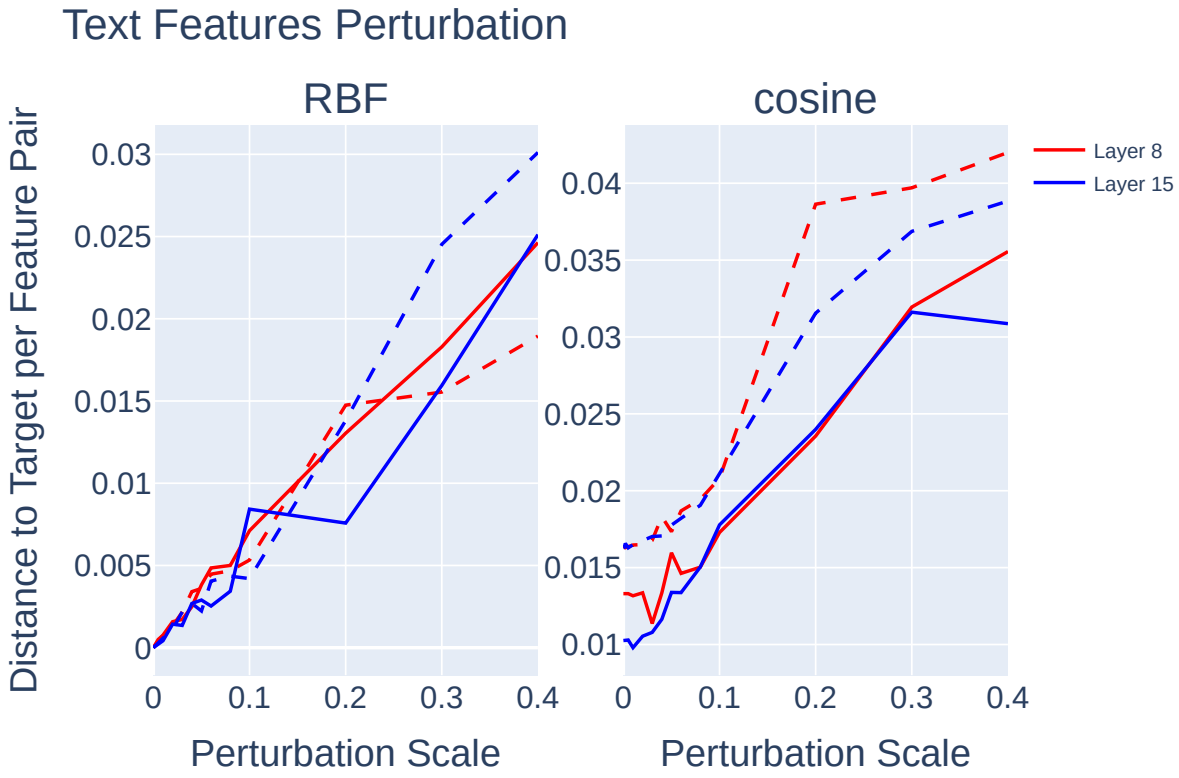

Figure 9: The dashed lines correspond to the permuted feature set. The plot shows how varying the perturbation scale (see subsection 4.2) changes how far the geometry of the abstract features diverges from the geometry of the target features. As the perturbation scale approaches 0, for the real features for both RBF and cosine similarity functions, the distance to the target PSM continues to decrease. This further confirms that the target geometries are energy minima. However for the RBF kernel, the permuted and unpermuted features exhibit roughly the same behaviour, indicating that the RBF kernel is not a good similarity metric, as it cannot distinguish between actual features and permuted ones. This may be due to the dependence on the norm of the features. In contrast, cosine similarity demonstrates a clear difference between permuted and unpermuted features.

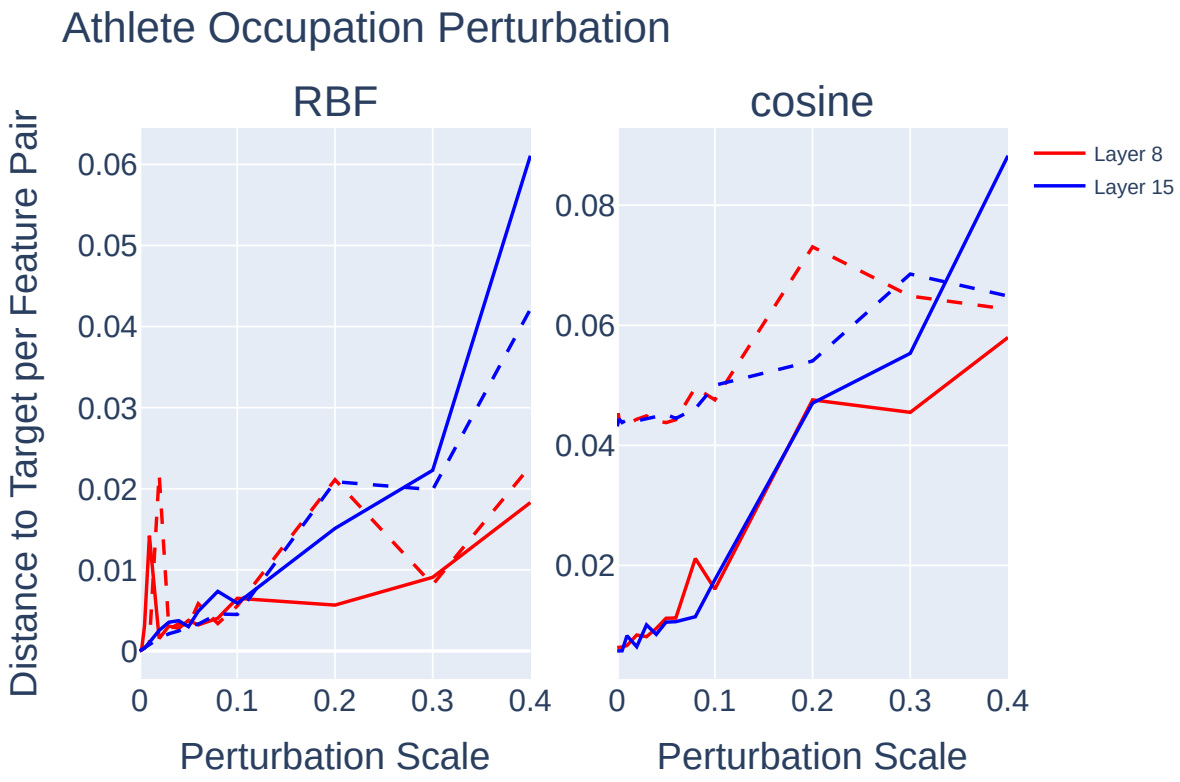

Figure 10: The dashed lines correspond to the permuted feature set. The plot shows how varying the perturbation scale (see subsection 4.2) changes how far the geometry of the abstract features diverges from the geometry of the target features. The story is largely the same as in Figure 9. There is a very clear distinction between permuted and unpermuted features in the cosine similarity case, and not in the RBF case. For small perturbations, the cosine similarity analysis would conclude that geometries associated with random features are not energy minima and geometries associated with actual features are (or at least that actual features are closer to energy minima than permuted features are).

5 Conclusion

We have seen several interesting phenomenon that point to non-trivial geometry in the activations of natural language Transformers. We have explored a method for analyzing the geometry of features and discovered some of its pitfalls. Our results are summarized as follows:

1. Groups of context and knowledge-dependent features related to facts about famous people seem to be arranged in simplexes. This holds true regardless of layer and is non-trivial given that random features in high-dimensional space are likely to be orthogonal.
2. For all groups of features of analyzed, geometry is established early on, and is then mostly preserved throughout all layers for norm-invariant similarity metrics. This could be because the features analyzed were too simple, but may also indicate a general issue with the feature extraction method.
3. The main claim, namely that features in actual Transformers are in energy minimizing configurations, seems to be largely true. Finding low-dimensional, abstract representations of the actual features that share the same geometry as them, and then optimizing for the energy leaves the geometry largely untouched. Furthermore, when the actual features are replaced by random features with the same per-dimension distribution, we find that they are not minima of the energy function, indicating that the method can distinguish between actual features and fake ones.
4. Cosine similarity is the metric of choice when compared to RBF similarity. When features are permuted and geometric structure is destroyed, we still find that abstract features matching the (permuted) target geometry are energy minima, which is not the case for cosine similarity.

One area that may indicate the geometry analysis method is lacking is in the feature importance weights. As discussed, they are typically ignored and set equal to each other in the joint optimization step. This either indicates that all features in each group were of equal importance, that the joint optimization procedure fails to take advantage of the benefit importance weights can give, or that our initial hypothesis was too simplistic.

5.1 Future Work

Deeper investigation into the energy minimization procedure is needed. While regularization is necessary for the re-optimization step, it appears fraught with difficulties, with results being highly sensitive to the choice of optimizer, λ_x, and norm constraint. It also complicates the claim that our abstract features are local minima of the energy function, since we are imposing arbitrary restrictions on norms that may not be present in an

actual Transformer. In addition, one of the main assumptions of this method, which is likely to be untrue, is that we have "all" the features in a given configuration when we do our geometry analysis. Suppose there was a set of 6 features in uniform superposition in 3 dimensions, so that they were in an octahedral configuration. If however, we mistakenly only choose 5 of those features to do our geometry analysis with, we will find that the configuration is not energy minimizing, since removing one of the vertices of the octahedron would create some "extra space" for the other features to take up that the target PSM would show they do not currently take up. It is clear that there is some relationship between the actual feature dimensionality, the number of features, and the dimensionality of our abstract features that remains to be explored.

One of the most common ways to validate an interpretability method is to check what results it produces for models with randomized weights [1]. Doing this for our method, while it was out of scope due to computational constraints for this work, would go a long way towards reducing the likelihood that this method provides hallucinatory results. In addition, the scale of feature sets and models in this work is quite small. In order to further evaluate it, a more comprehensive evaluation across a broader class of features, feature extraction methods, and models would be necessary. Finally, it may be possible to use this general framework to do more interesting things such as identifying the correlation structure between features. It may be that the PSMs themselves give some information about this; for example, duplicate features would be highly correlated and would have a large similarity between them. Or, if we take it as ground truth that features are in energy minimizing configurations, is it maybe possible to identify features that are "missing" from a given configuration? One could identify these "missing" features in abstract feature space, and then find tokens in the actual dataset that strongly activate the hypothesized feature. This could be a way to figure out exactly which features are in superposition with what other features, and bootstrap feature extraction in general.

Acknowledgements

I would like to express my gratitude to my wonderful supervisor Professor Vardan Papyan who has been a joy to work with these past several months, providing valuable insight and always being encouraging and helpful.

I would also like to thank my family, friends, and professors who have helped me get through these past 4 years of Engineering Science and this thesis journey. I have grown so much as a researcher, learner, and person, and I am excited to apply the skills I've learned to make the world a better place in whatever way I can. It was almost never easy, but I truly believe that this has been and will be one of the most enriching experiences of my life, and I wouldn't trade it for the world.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps, 2020.

[2] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy, 2018.

[3] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

[4] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

[5] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[7] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.

[8] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17: 449–467, 1965. doi: 10.4153/CJM-1965-045-4.

[9] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022.

[10] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020.

[11] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories, 2021.

[12] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

[13] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing, 2023.

[14] Weiyang Liu, Rongmei Lin, Zhen Liu, Lixin Liu, Zhiding Yu, Bo Dai, and Le Song. Learning towards minimum hyperspherical energy, 2020.

[15] Weiyang Liu, Longhui Yu, Adrian Weller, and Bernhard Schölkopf. Generalizing and decoupling neural collapse via hyperspherical uniformity gap, 2023.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[17] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models, 2023.

[18] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2023.

[19] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

[20] Chris Olah, Nick Cammarata, Chelsea Voss, Ludwig Schubert, and Gabriel Goh. Naturally occurring equivariance in neural networks. *Distill*, 2020. doi: 10.23915/distill.00024.004. https://distill.pub/2020/circuits/equivariance.

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles

Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[23] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[24] BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vas-

silina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush

Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. Bloom: A 176b-parameter open-access multilingual language model, 2023.

[25] Zeyu Yun, Yubei Chen, Bruno A Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors, 2023.

A Permutation Finding Algorithm

Our algorithm to approximately find the permutation P in Equation 2 is split into two steps, sorting and greedy swapping. To save recomputing all entries of the PSM every time a new permutation is tried, we instead permute T directly, by appropriately rearranging the rows of T to give the same result as if we had permuted the order of the features that was used to generate T. This algorithm may be improved by turning to classic problems in computer science such as the graph isomorphism problem or the Blossom algorithm [8].

A.1 Sorting

Since applying the permutation P to T involves permuting its rows, one approach would be to map each row of T (and S) to a scalar, and then sort based on those scalars. This map $f : \mathbb{R}^N \to \mathbb{R}$ should be permutation invariant, since when a permutation is applied to T, its columns are permuted as well. This would mean a given row would have its entries permuted. f should produce the same output in the order space. f should also have the property that two rows t_i, t_j of T with similar *sets* of entries should get mapped to similar values and two rows with dissimilar *sets* of entries should get mapped to different

values.

One suitable f is as follows:

$$f(t_i) = \sum_{p=1}^{p_{\max}} ||t_i||_p, \tag{6}$$

where $||\cdot||_p$ denotes the p-norm, and $p_{\max}$ is some constant, positive integer. In practice, $p_{\max} = 5$ works well.

Sorting T and S using this method gives us two permutations P_T and P_S. If we assume that T and S are actually permuted versions of each other, we can then reconstruct the permutation to get from T to S by the following:

$$\begin{aligned} P_T \circ T &\overset{\text{want}}{=} \hat{P}_S \circ S \\ \implies P_S^{-1} \circ P_T \circ T &= S, \end{aligned}$$

allowing us to recover the permutation $P_S^{-1} \circ P_T$ that takes T to S.

A.2 Greedy Swapping

While the sorting step typically gives us something very close to optimal, we apply one additional step to further optimize how close T is to S. This step involves enumerating all possible pairwise swaps between rows, computing the Euclidean distance between T and S after the swap, and then applying the one that reduces the distance the most at each step. Note that this is done to a version of T that has had the permutation from the sorting step applied to it already. If the distance cannot be decreased, or if the matching is already perfect, the algorithm terminates. Typically, around 1 or 2 swaps (but most often 0) are applied in this step, since sorting brings us close to the optimal solution already. If we run greedy swapping without sorting, we typically apply many more swaps and converge to a worse solution.