Covariance, Whitening, and the SVD

A Self-Explainer for Neural Network Activations

Sheridan Feucht [1]

Whitening has reared its head in a number of interpretability papers, including LEACE, ROME, LRE, and the LRH paper. What exactly is it? What does it mean to whiten a vector space? Can we relate this to SVD and the intrinsic dimension of a representational space? Although weights of a model look full-rank, if we have a squished enough space it might be the case whitening really matters.

A self-explainer is a document that is written by Sheridan to explain a specific concept to Sheridan. You can read it if you want. If so, I hope it is clear for you as well.

1 Activations vs. Weights

It's important to remember that SVD is just a way of decomposing matrices, akin to breaking down a number into its factors. In this self-explainer, we are talking about SVD specifically in the case of its applications to *data* (i.e. doing PCA on model activations). You can also apply SVD to model *weights*, but just because it's SVD, that doesn't mean it's inherently connected to what we're doing here.

2 Prerequisite: Covariance Matrices

Let's say you have a collection of activations in a neural network, collected in the residual stream after a particular layer. Each of these activation vectors $\boldsymbol{x}$ is a point in our thousand-dimensional representation space, and each dimension of that vector can be thought of as a random variable that has some mean and variance.

When we talk about the *covariance matrix* $\boldsymbol{C}$ of activations at that layer, we're looking at how much every dimension co-varies with every other dimension. For example, the covariance between the third feature and the 309th feature would be represented by entries (3, 309) and (309, 3), as $\boldsymbol{C}$ is symmetric. In a world where features are neuron-aligned, you might imagine a high covariance between

Affiliation [1] Northeastern University
Correspondence sheridan.feucht@gmail.com
Version January 12, 2026

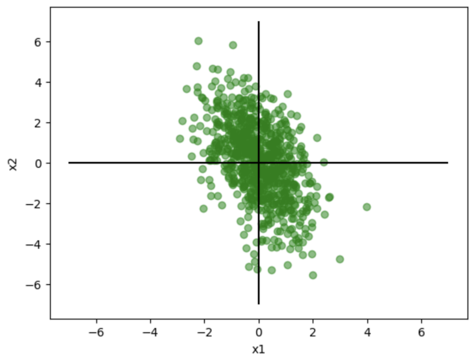

Figure 1: Our $n = 100$ random sample of 2D vectors.

dimensions representing, e.g., gender and income levels. It's almost exactly the same thing as correlation, just unnormalized.

According to Wikipedia, if we represent the activation space as a column vector with random variables in each entry $\boldsymbol{x} = (X_1, X_2, ..., X_n)^T$, then the covariance matrix is defined as

$$\boldsymbol{C} = E[(\boldsymbol{x} - E[\boldsymbol{x}])(\boldsymbol{x} - E[\boldsymbol{x}])^T]. \tag{1}$$

But how do you actually calculate this in real life? We don't know the true distributions of each random vector. We have to go based off some sample of activations from the model. Let's construct a matrix $X \in \mathbb{R}^{n \times d}$, where d is the model dimension and n is the number of activations we've collected from model internals, over a large Pile of text. For our little example, I'll generate points with $d = 2$, so I can plot things easily. In Figure 1, each point is a two-dimensional activation vector inside a made-up model.

```python
mean = points.mean(axis=0) # points are shape (n, 2)
centered = (points - mean).T # transpose into column vectors
np.matmul(centered, centered.T) / (len(points))
```

To calculate the covariance of the model's activations based on this sample, we use the above lines of code. We calculate the sample mean of the vectors given to us and subtract that off of each example to center the data (which doesn't matter too much here, as it's already centered around the origin). Then, we transpose `centered` so that the examples are all column vectors. Multiplying this with its transpose can be seen as taking the outer product of every example with itself, giving us a sum over n rank-one 2x2 matrices. In other words, we're calculating

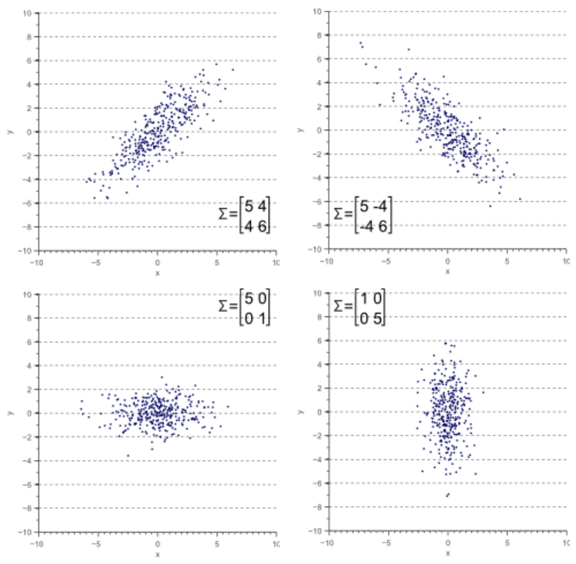

Figure 2: Figure borrowed from `https://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/`.

the inside part of Equation 1 for each example. Summing these all together and dividing by the number of examples gets us the outer expectation seen in said equation. Doing this on the points in Figure 1, we get

```
array([[ 0.94713418,  -0.87512289],
       [-0.87512289,   3.4349716 ]])
```

for $n = 800$ examples. These numbers are pretty close to the covariance matrix I actually used to generate the data! (I used a multivariate normal distribution with zero means and covariance matrix `[[0.9, -0.8], [-0.8, 3.5]]`).

3 Geometric Understanding of Covariance Matrix

The diagonal entries of a covariance matrix are just variance, so they describe the spread over each axis. The off-diagonal values describe the "orientation" of the data, shown in Figure 2. This article (link here) details how we might want to represent the covariance matrix with a single vector v pointing in the direction of most variation.

4 Covariance, PCA, and SVD

The previous section should seem really familiar, because it's the first step in doing Principal Component Analysis (PCA). After we've calculated the covariance matrix of our activations, we can compute the eigenvectors of $\boldsymbol{C}$, which become our principal components. The main takeaway from this section is that PCA, which is based on the eigenvectors of a dataset's covariance matrix, is almost exactly the same thing as SVD. This connection hinges upon the fact that **covariance matrices are symmetric**, since $\text{Cov}(x, y) = \text{Cov}(y, x)$.

4.1 PCA Relates To SVD

PCA should really remind you of Singular Value Decomposition (SVD), but it's a little hard to see how these two things are related at first blush. In a nutshell, the connection is that *(1) PCA uses the eigendecomposition of* $\boldsymbol{C}$ $\to$ *(2)* $\boldsymbol{C}$ *is symmetric* $\to$ *(3) eigendecomposition* $\approx$ *SVD for symmetric matrices.*

(1) PCA uses the eigendecomposition of $\boldsymbol{C}$. In eigenvalue decomposition, given a matrix A, we look for eigenvectors Q and eigenvalues Λ such that $AQ = \Lambda Q$, where Λ is diagonal. If there are enough linearly independent eigenvectors to form an invertible matrix Q (which is true if all the eigenvalues are unique, and sometimes even when they're not), then A is *diagonalizable,* which means that we can express it as $A = Q\Lambda Q^{-1}$.

(2) $\boldsymbol{C}$ *is symmetric.* As alluded to above, if A is symmetric, then we get the special properties that its eigenvalues are real and its eigenvectors are all orthogonal to each other. Thus, every symmetric A is diagonalizable, with $A = Q\Lambda Q^{-1} = Q\Lambda Q^T$ (the spectral theorem). You can think of any symmetric transformation as a rotation into the eigenvector basis (Q^T), some stretch along those basis eigenvectors (Λ), and then a rotation back into the standard basis (Q).

(3) Eigendecomposition $\approx$ *SVD for symmetric matrices.* If you are familiar with the SVD, this should be ringing a bell! Indeed, for symmetric matrices, the eigenvalue decomposition $A = Q\Lambda Q^T$ is almost the same as the SVD $A = U\Sigma V^T$. This is because if A is symmetric, then $U\Sigma V^T = (U\Sigma V^T)^T = V\Sigma U^T$ which implies that $U = V$ and $A = U\Sigma U^T$, just like the spectral theorem. The only difference is that singular values σ along the diagonal of Σ have to be non-negative, whereas eigenvalues in Λ are allowed to be negative. All that means is that we have to flip the signs of the eigenvectors around, and then the representations are "the same."

Therefore, we can say that the eigendecomposition of $\boldsymbol{C}$ that we use in PCA is the same thing as the SVD of $\boldsymbol{C}$ (up to signs).

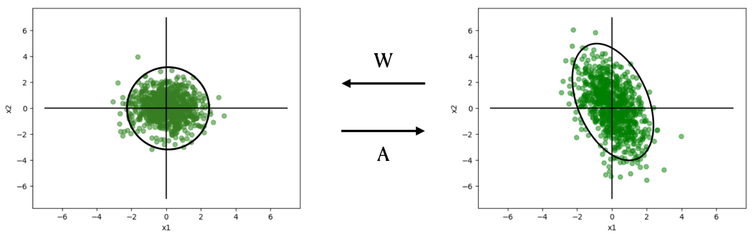

Figure 3: Whitening transformation.

4.2 Why is Covariance Used in PCA?

5 Whitening

If the covariance matrix $\boldsymbol{C}$ describes the covariance of each dimension with each other dimension, then the *whitening transformation* W is inverse of A, the matrix that transforms the unit sphere into an ellipsoid with covariance structure $\boldsymbol{C}$ (Figure 3). If we can work out what A is, we can invert it to remove the covariances in our activation data!

5.1 Derivation

First, we have to figure out what A is in relation to $\boldsymbol{C}$. Following the conventions in Trefethen and Bau's Numerical Linear Algebra textbook, let's look at an arbitrary unit vector v in the whitened space (left in Figure 3) and its corresponding transformed version $u = Av$. Using the definition of covariance, we have that

$$\boldsymbol{C} = E[uu^T] = E[(Av)(Av)^T] = E[Avv^TA^T] \tag{2}$$

$$\boldsymbol{C} = AE[vv^T]A^T = AA^T \tag{3}$$

The last step is possible because $E[vv^T] = I$, as the v vectors are in the whitened space and only co-vary with themselves (and are unit vectors). Therefore, $C = AA^T$. If we want to find a whitening transformation W, all we need to do is break $\boldsymbol{C}$ into invertible matrices AA^T, then take $W = A^{-1}$ as our whitening transformation. But how do we do that factorization?

5.2 Types of Whitening

There are a number of ways of choosing A so that $\boldsymbol{C} = AA^T$. You can take the Cholesky decomposition, calculate the SVD of the covariance matrix (PCA whitening), or add an extra rotation to that to get ZCA whitening.

Cholesky decomposition. Because $\boldsymbol{C}$ is symmetric and positive semidefinite, we can decompose it into the product of a lower triangular matrix and its conjugate transpose, $\boldsymbol{C} = LL^*$, where diagonal entries are allowed to be zero. Generally, this isn't as stable and universal as the PCA-based methods we'll discuss next.

PCA Whitening. Another approach to whitening is to just use the SVD to break down $\boldsymbol{C} = U\Sigma U^T$ (since $\boldsymbol{C}$ is normal, its left and right singular vectors are equal). We can take the matrix square root of Σ to factorize $\boldsymbol{C} = (U\Sigma^{1/2})(\Sigma^{1/2}U^T) = (U\Sigma^{1/2})(U\Sigma^{1/2})^T$ where now we have a whitening matrix $W = (U\Sigma^{1/2})^{-1} = \Sigma^{-1/2}U^T$.

ZCA Whitening. Given the decomposition above, we can tack any arbitrary rotation onto the end of each matrix $U\Sigma^{1/2}R$ and still get the same factorization of $\boldsymbol{C}$. If we strategically choose $R = U^T$, then our whitening transformation becomes $W = U\Sigma^{-1/2}U^T = \boldsymbol{C}^{-1/2}$. It's just the same as PCA whitening, except we rotate everything back into the standard basis afterwards with U. This is also called Mahalanobis whitening. This post has a great explanation of ZCA vs. PCA.

5.3 Whitening In Papers

There are a few papers where whitening naturally falls out of the derivations the authors get. In ROME (Meng et al., 2022), they get a term KK^T in their derivations, where K is a matrix with 100k samples of key vectors as its columns. You can see that then KK^T would be an uncentered version of Equation 1. To do "whitening," they actually just take the inverse of KK^T itself, instead of square-rooting it like in ZCA. (I'm not sure why $\boldsymbol{C}^{-1}$ only shows up once in the rank-one update; wouldn't you need to unwhiten the whole update before adding it to the weights?)

In LEACE (Belrose et al., 2023), they use Mahalanobis/ZCA whitening, taking the square root inverse of the covariance matrix. Figure 4 shows their erasure process including whitening. In this particular paper, the red line represents the column space of the cross-covariance matrix between activations and class labels. Imagine that red is the "gender subspace," and green is the "income subspace." Projecting directly onto the dashed line would mean that we would destroy a lot of the variation along the "income levels" subspace. If you whiten first and project onto the green subspace instead, you get less interference. Of course, the authors do not assume in this paper that the green line corresponds to a meaningful subspace; rather, they are just trying to move points around as little as possible. If we are to believe that orthogonality *means something* in whitened space, then we might interpret LEACE's whitening in that way.

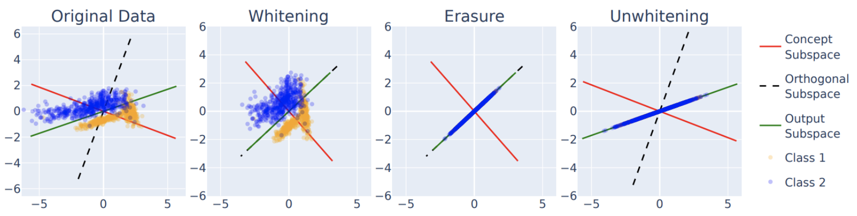

Figure 4: Figure 1 from LEACE. In the original space, you can see that there's a slight positive covariance between x and y. Their red line represents the column space of the covariance between the data and class labels. Instead of directly projecting onto the dashed line, they whiten the space and project onto the green line (which is orthogonal to red in whitened space), so as to move the points around as little as possible.

This example also provides context for why Sam Mark's mass-mean probing approach seems to work. In Marks & Tegmark (2024), he argues for taking the difference between the means of two classes to get the concept direction for that class (which is the red line), instead of looking at decision boundaries found by logistic regression. This connection makes sense, because LEACE is all about trying to make class-conditional means equal.

Finally, the linear representation hypothesis paper (Park et al., 2024) does a bunch of math that I still don't fully understand, but ends up at a "causal inner product" that just does whitening in the output embedding space based on the covariance of all of the rows of the language modeling head. In other words, they take the `lm_head` with dimensions (`vocab_size`, `model_dim`) and take the centered covariance across all of those rows. This seems like it empirically works based on their experiments; things that intuitively seem like they should be orthogonal become orthogonal.

6 Extra Notes

6.1 Approximately Low-Rank

If the covariance between two dimensions is extremely high, you can imagine that the unit sphere becomes so flattened out that it's basically like a sheet of paper. In such a case, this space could be thought of as being "approximately low-rank", since variation orthogonal to that squished dimension would be negligible. These tiny singular values would be better treated as just zero. This idea also makes sense if you think about the vector dimensions themselves–if two dimensions covary to an extreme degree, that means that they're essentially the same value all the time. Clearly, in such a case, the rank of that representation would be 1 dimension lower.

6.2 Data versus Fan-Outs

One important note from David is this: whereas ROME and LEACE describe the covariance structure of activation space over many inputs, the LRH paper gets its covariance matrix just from the language modeling head. This has really interesting implications, because it means that we might be able to do the same thing for all of the fan-outs that appear throughout LLMs (i.e. MLP fanout layers).

7 Future Rabbit Hole: CKA, HSIC, and ICA

So, look at where else covariances matrices come up... in something you might find familiar, centered kernel alignment (CKA). The definition of CKA actually is just an adjusted version of the Hilbert-Schmidt Independence Criterion (HSIC). It takes in two matrices that are assumed to be kernels, and measures the similarity between them.

$$\mathrm{CKA}(K, L) = \frac{\mathrm{HSIC}(K, L)}{\sqrt{\mathrm{HSIC}(K, K)\mathrm{HSIC}(L, L)}} \tag{4}$$

But what is HSIC? I still don't really know. And what's crazy is that when I look it up, ICA comes up, which is another thing I don't know. But I'm going to leave this rabbit hole for another day.