Projective Dictionaries

Michael Sklar

October 2024

1 Introduction

This document proposes the projective dictionary for neural network interpretability, discusses technical background, and develops use cases to aid two methods: the Sparse Auto Encoder (SAE) and Geiger et al's Distributed Alignment Search (DAS).

2 Introduction

DAS and SAE's are two different optimization techniques used for neural network interpetability. They are used for quite different purposes, with DAS seeking causal structure from a defined hypothesis and SAEs seeking unsupervised featurization; however both methods are used with the intention of isolating directions in the network's activations and identifying them with interpretable concepts.

One difficulty faced by these methods is that the number of features to consider is potentially far greater than the d_{embed} dimension in which the optimization operates. This problem, particularly if it occurs for some set of 'true features', is sometimes referred to as "superposition" [3]. For DAS, superposition and generally the question of how to handle large classes of features poses an unsolved problem, although work on this topic is ongoing with the development of HyperDAS (not yet released). In the SAE literature, several approaches to the superposition issue have been used, including top-k filtering for the k most active latents [5], and L-1 penalties in conjunction with a threshold operation such as RELU or jump-RELU [7].

Once a dimensional subspace is identified for DAS, it defines a reconstruction of interest according to a projection, preserving norms in the identified "feature subspace" (referring to the span of the first $d_{intervention}$ rows of the rotation matrix R which form the intervention space of DAS). In contrast, the SAE approach, which typically uses linear encoding, followed by a sparsification operation followed by linear decoding, is meant to be computationally quick-per-sample, especially compared to classic methods such as Method of Optimal Directions (MOD) [4]. Both the MOD and SAE objectives aims to reconstruct the input activations, but unlike MOD, the SAE may struggle to reconstruct the correct signal-strengths even within the span of the identified feature directions. [Here we use the convention of terming the decoder vectors as the "feature directions.", in accordance with [1].]

Another type of difficulty that arises for SAE's is "feature splitting" [6] and "feature absorption" [2]. A recent [2] blog post claims that feature absorption is reduced by tying the encoder and decoder directions. However, (I am inferring on weak evidence and don't actually know / haven't looked at this on real cases) due to correlations between nearby features this tying affects the reconstruction quality, so this approach has not been used in the major works.

2.1 Projective Dictionaries

The idea of projective dictionaries is simple to motivate: we would like to use a SAE-like mechanism for selection, but instead of outputting a direct additive reconstruction, we (either approximately or exactly) project the target onto the subspace spanned by a few selected features.

Our hope is that projective dictionaries can resolve several issues mentioned in the above section, offering:

1. Reduce dimensional-footprint for DAS / Hyper-DAS, hence, better-quality feature isolation and more interpretability

2. Scalability of DAS / Hyper-DAS to work with wide ranges of features and entities

3. Improved reconstruction quality of SAE "variance explained" for a given sparsity of the feature-set

4. (maybe) Tolerability of tying the SAE encoder and decoder weights (tbh I don't actually know what was unacceptable about that in the first place, but my guess is that this introduced reconstructive error due to correlations)

5. (highly speculative) this might also improve the dictionary and reduce the incentive to perform feature-absorption / feature-splitting by allowing better performance with relatively few features

A version of this projective dictionary can also be used for the cross-layer "transcoder" version of SAEs.

3 Background: linear regression

What does it mean to project onto the set spanned by an identified set of features?

This material is likely to be familiar for those with graduate-level exposure to linear and ridge regression with matrix operations.

The linear regression problem is, given target to predict $Y : n \times 1$ with data $X : n \times p$,

$$Minimize_B : ||Y - X\beta||_2^2$$

This can be solved by expanding the square as

$$||Y - X\beta||_2^2 = (Y - X\beta)^T(Y - X\beta)$$

And then seeking the minimum via finding the critical point where derivatives with respect to each component of β_j are zero.

Taking these derivative and setting them to zero, and rearranging, yields the *normal equations*, which we write suggestively as:

$$X^T(X\beta) = X^TY$$

If we write $X\beta$ as $\hat{Y}$, then this tells us:

$$X^T\hat{Y} = X^TY$$

in other words, that our reconstruction must have the same dot-products with each column of X, as Y does.

As another way of saying: consider any reconstruction procedure that (1) lies within $span\{X\}$ and (2) imposes that the reconstruction keeps all the same dot products with each X_j as Y had. Then if X is full-rank, that procedure must be identical to performing regression of Y on X.

Returning to the derivation:

With X full-rank, the normal equations can be solved as

$$\beta = (X^TX)^{-1}X^TY$$

which yields the prediction/reconstruction

$$\hat{Y} = X\beta = X(X^TX)^{-1}X^TY$$

Notice that the prediction is linear in Y. We can collect this "quadruple-X" expression as a matrix H, yielding

$$\hat{Y} = HY$$

Notice that H is a projection matrix: $H^2 = H$. Its eigenvalues are all either 0 or 1. It preserves everything inside of $span\{X\}$ and kills off everything else.

Consider now the computation. $(X^TX)^{-1}$ can be numerically unstable if X is nearly singular. We discuss fixes to this with regularization in subsequent sections. However, note also that the normal equations are often used iteratively to seek β which solves the normal equations, which then allows us to skip the matrix-inversion entirely by simply returning the solution for β or $X\beta$.

4 The Tied-Projective-Dictionary SAE

With the above in-hand, we are now already able to define a simple version of an SAE procedure, using this projection mechanism.

Let's decide to simplify our dictionary by enforcing that the encoder directions are the same directions we'll use for feature-projection (we'll undo this restriction in later sections).

Our dictionary is too large to want to use every feature so we'll have to select down.

Then, how do we decide which ones to use?

A natural way is to use the selection process of the top-k Sparse AutoEncoder.

That is, for each dictionary element we take its directions f_j, we dot-product against Y, optionally add a feature-specific bias c_j, and then we do a top-k selection, yielding

$$\mathit{selection} := \text{Topk}\left\{F^T Y + c\right\}$$

Then, we subset to our selected columns, setting $X := F[:, \mathit{selection}]$, and compute the regression reconstruction. The loss is then:

$$||Y - \hat{Y}||_2^2 = ||Y - X\beta||_2^2 = ||Y - X(X^T X)^{-1} X^T Y||_2^2$$

It may be recommended to skip inversion, via e.g. solving for β with normal equations. If X is full-rank, this process should be possible to backpropagate.

Will this form solve the various issues that have been mentioned for SAE's, reducing feature splitting? In particular, there is a trainability concern that is worth highlighting here, which we will solve in later sections. For now we will merely describe the issue: consider that the top-k selection function does not pass any gradient signal, so no gradients reach c_j. Thus,uUnlike the typical SAE, which makes use of the fact that the encoder differentiably affects the reconstruction signal (and in fact the SAE is quite intentionally constructed this way, with the decoder normalized so that the encoder's dot-product directly implies the signal-strength), here we cannot learn c_j - it simply will not move after initialization.

We will return to fix this issue after developing some more mathematical background.

5 Ridge regression

As mentioned in the previous section, several issues can arise with regression if X is nearly singular (not to mention if X has more columns than rows!)

In this case, a classic way of resolving the issue and ensuring that the matrix will invert, is to use *ridge regression*, which adds a penalty to the loss proportional to the sum-of-squares of β:

$$||Y - X\beta||_2^2 + \lambda||\beta||^2$$

What does this do to the fit? It turns out that it makes the matrix inversion much better.

The ridge regression solution is:

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T Y$$

where I is a $p \times p$ identity matrix.

Notice that when we set $\lambda = 0$, we recover the linear regression solution.

However for any positive value of λ, the addition of λI ensures that the matrix to be inverted has positive eigenvalues bounded away from zero, and thus is always invertible. Furthermore, it ensures continuity and differentiability of the regression fit, predictions, and loss with respect to the feature matrix X. These are very desirable properties.

Using this penalty also means that the resulting regression matrix,

$$H_\lambda := X(X^T X + \lambda I)^{-1} X^T Y$$

is no longer a projection matrix, i.e.,

$$H_\lambda^2 \neq H_\lambda$$

unless $\lambda = 0$. Its eigenvalues are 0 for eigenvector outside of the span of X, and less than 1 for eigenvectors inside the span of X.

This means that the reconstruction $\hat{Y}$, even if Y does lie in the span of X, is not going to be perfect. And the reconstruction loss is going to be greater, the larger we drive λ.

In the next section, we will try to use this effect as way to get signals for variable importance : we will parametrize λ not as a scalar, but rather as a vector so that we can have a sub-penalty component λ_j for each feature-index j. Those feature dimensions which are penalized-least, will be allowed to have more-accurate reconstructions in their span. Hence, if we hook up our encoder's importance scores to determine which penalty is used, we can get back signals about which dimensions were more and less important, and use this to perform top-k selection.

[Note: by the way, performing top-k selection at all is actually technically optional, now that we are doing ridge regression! It is possible to keep the whole dictionary in. However, to prevent degenerate overfitting, it would be necessary to have very strong penalties on most coefficients so that only a few dimensions can influence the reconstruction. And at that point, for computational speed reasons, we might as well simple axe the least important feature dimensions.]

Given that the ridge penalty makes dimensions not-completely-active, what's the right way to measure effective dimensionality?

One way to give an answer to this is to measure, how sensitive are the predictions $\hat{Y}$ to isotropic noise in Y? If you asked this question for linear regression, you'd get an answer which linearly grows with the dimension.

This measure is sometimes called "degrees of freedom" or "effective dimensionality." It can be computed for ridge regression. The result is, according to

https://datavis.ca/papers/genridge-jcgs.pdf ,

of the form:

$$\sum_j \frac{d_j^2}{d_j^2 + \lambda}$$

where d_j are the singular values of X.

6 Encoder-Decoder Projective Dictionaries (for HyperDAS and Transcoders)

Now, using the derivations from the previous section, we can complete the analogy to how top-k sparse auto-encoders used a measure of predicted signal-strength to select dimensions.

We shall now generally allow for the encoder matrix E to be not necessarily the same as the decoder matrix.

We will forcibly normalize all of the columns f_j in our dictionary decoder matrix F.

We shall also now consider the possibility that the input to the encoder, denoted matrix A, is not necessarily the same as the reconstruction target Y. This permits us to use this technique for transcoders, where A is the activations of the prior layer, or for the HyperDAS architecture, where A is from the activations of the editor hypernetwork.

We compute the encodings with a matrix-multiply plus bias, yielding a vector e computed as

$$e_j = E_j.A + b_j$$

These e_j's will be used as importance scores; in analogy to the top-k SAE, we wish to use them to select the relevant features (winnowing to some top-k, perhaps as well as omitting those with negative scores).

The question remains: how to connect these to ridge regression in a feature-specific way, such that we can connect a penalty to each feature, and such that the e_j's can be learned to represent feature importance?

Consider the following generalization of the ridge regression formula:

$$\hat{Y} := X \left(X^T X + \mathrm{diag}(\lambda)\right)^{-1} X^T Y$$

where λ is no longer a constant but now a vector, with $|E|$ different entries. One way to interpret this added flexibility is to notice that increasing a λ_j increases the regression's penalty on feature f_j, dampening the amount of projection and leaving more signal on the table in that dimension.

We would like connect this back to the encoder mechanism. As a convention, we would like for the encoded latent score e_j to signify feature importance, hence a smaller associated penalty, whereas a larger λ_j signals decreased featue importance.

We may approach this by mapping $e_j \to \lambda_j$ as a decreasing function over positive e_j and λ_j, such as $\lambda_j = e_j^{-2}$, where it is assumed that features with $e_j \leq 0$ are omitted.

With this connection made, we can now proceed to apply this feature-specific ridge-reconstruction, using the encoder directions to select features, and using that subset of columns of F to form the matrix X used to reconstruct the target Y via the regression formula. This yields a reconstruction whose MSE loss can be backpropagated, now reaching both the encoder and decoder directions. (Recall that the decoder columns must remain normalized to 1).

One final issue still remains: in this parametrization, we have incentivized the optimization to tell us that all features are highly "important" and deserving of minimal ridge penalty.

We have not put a quota on the amount of importance, and so the norms of E's columns will race continuously upward as we train.

In order to prevent this, we have a few options, including:

- Normalize or fix the total sum of importance scores or the λ_j's
- Penalize feature importance activity or activity of the λ_j's

It is not obvious which of these is preferable, as they are all very closely related procedures.

We can, however, look back to our previous material for inspiration, seeking a convenient notion of dimensionality to fix or penalize. Recall that the effective dimensionality of ridge regression with constant λ is written as

$$\sum_j \frac{d_j^2}{d_j^2 + \lambda}$$

where d_j are the singular values of X; and in fact, each term in this sum corresponds to an eigenvalue of the H_λ matrix.

While the following formula is not actually an accurate application of this equation once we generalize to feature-specific λ_j unless X's dimensions are non-orthogonal, we can hope that it will be sufficiently closely, and perhaps in a sense conservative, to estimate the effective dimension as

$$g(\lambda(A)) := \frac{1}{1 + \lambda_j}$$

.

[approximating the singular values as all 1, as if the dimensions of X are approximately orthogonal, noting that the columns of F and hence X all had norm 1. Even though the singular values of X aren't all 1 in generality, I expect this to be fine, and it's not a bad thing to punish redundancy of dimensions anyway.]

Then we can seek to either penalize this effective-dimension quality by tacking on $g(\lambda(A))$ to the penalty for backpropagation; or alternatively, we can consider using this effective-dimension computation as part of the top-k selection, by only selecting as many dimensions as will fit under a certain total budget of feature-influence.

Note, also, that this setup can now be used with a tied-decoder for the same-layer SAE setting, to solve the problem of the lazy biases in earlier section. Simply implement this architecture, and forcibly tie the encoder, and set A and Y as the same object. The result will use dampened projections instead of exact projection, but in exchange, can use the slack to fit importance scores.

7 Conclusion

This completes our description of the projective dictionary. In our view, it is proposed to be a principled evolution of the top-k sparse-auto-encoder, which swaps the additive reconstruction step for a subspace-projection reconstruction. As next steps, we encourage exploration and measurement of its utility for DAS and SAE's, according to the 5 candidate purposes mentioned in the introductory section.

References

[1] Trenton Bricken et al. "Towards Monosemanticity". In: *Anthropic* (Oct. 2023). Part of the Transformer Circuits Thread. URL: `https://transformer-circuits.pub/2023/monosemantic-features`.

[2] David Chanin et al. "A is for Absorption: Studying Feature Splitting and Absorption in Sparse Autoencoders". In: *arXiv preprint arXiv:2409.14507* (2024).

[3] Nelson Elhage et al. "Toy Models of Superposition". In: *Anthropic* (2022). Part of the Transformer Circuits Thread. URL: `https://transformer-circuits.pub/2022/toy_model/index.html`.

[4] Kjersti Engan, Sven Ole Aase, and J Hakon Husoy. "Method of optimal directions for frame design". In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258).* Vol. 5. IEEE. 1999, pp. 2443–2446.

[5] Leo Gao et al. "Scaling and evaluating sparse autoencoders". In: *arXiv preprint arXiv:2406.04093* (2024).

[6] Aleksandar Makelov, George Lange, and Neel Nanda. "Towards principled evaluations of sparse autoencoders for interpretability and control". In: *arXiv preprint arXiv:2405.08366* (2024).

[7] Senthooran Rajamanoharan et al. “Improving dictionary learning with gated sparse autoencoders”. In: *arXiv preprint arXiv:2404.16014* (2024).