
Recovering Goodfire’s SAE feature vectors from their API

Lovkush Agarwal
Independent

With
Goodfire and Apart Research

Abstract

In this project, we carry out an early trial to see whether Goodfire’s SAE feature vectors can be recovered using the information available from their API.

The strategy is: pick a feature of interest, construct a contrastive dataset using Goodfire’s API, then use TransformerLens to get a steering vector for the contrastive dataset, by simply calculating the average difference in the activations in each pair.

The early trial produced underwhelming results, however, I am confident one could reconstruct the underlying SAE feature vectors, with further experimentation and/or better techniques for obtaining the steering vector from a dataset.

Keywords: AI Observability, Mechanistic Interpretability, Model Reprogramming

1. Introduction

Goodfire’s API does not expose the underlying SAE feature vectors, but instead provides various convenient methods that are all directly built on top of these feature vectors. Naively, these SAE feature vectors are a key part of Goodfire’s value proposition, because training SAE’s is generally considered a challenging and expensive task. Once you have the vectors, identifying which features are firing for each token or steering based on the features is straightforward.

The question arises: can these SAE feature vectors be recovered based on the information available through the API.

During this hackathon, I implemented one simple strategy to try to extract the vectors. The results were not promising, but I am confident one could reconstruct the underlying SAE feature vectors, with further experimentation and/or better techniques for obtaining the steering vector from a contrastive dataset.

2. Overview

In detail, the strategy is as follows.

1. Pick a feature whose vector we want to extract.
 - E.g. ‘betrayal and treachery’
2. Create a contrastive dataset consisting of triples (p_i, x_i, y_i) .
 - The p ’s are prompts, the x ’s and y ’s are created using Goodfires API so that the x ’s are nudged to have the feature reduced and the y ’s are nudged to have the feature increased.
 - E.g. p is ‘I am playing a game of diplomacy. What advice do you have for me?’
 - E.g. x is ‘Diplomacy! A game of strategy, negotiation, and cunning. Here are a few tips to help you navigate the world of Diplomacy...’
 - E.g. y is ‘A knife in the back, a stab in the dark... how can I trust you? As a friend, I’d advise you to keep your eyes open and your tongue sharp. Don’t be afraid to make deals and forge alliances, but never underestimate...’
3. Use the contrastive dataset to determine a steering vector. I used the following simple strategy:
 - Pass (p_i+x_i) and (p_i+y_i) through a model and cache the activations
 - Calculate the difference in the activations, for the first n tokens after p_i .
 - i. In my experiments, I used $n=1$.
 - The steering vector is the average of all these differences, across n and across the dataset.

To evaluate the end result, I tried using the final vector to steer the LLM on the same prompts. The results were not promising and did not show much, if any, signs of ‘betrayal and treachery’. For example:

- p is ‘I am playing a game of diplomacy. What advice do you have for me?’
- Unsteered generation: ‘Diplomacy! A game of strategy, negotiation, and betrayal. Here are some tips to help you navigate the complex world of diplomacy’
- Steered generation: ‘Diplomcy! A game of strategic alliances, cunning diplomacy, and... occasional betrayal. Here are a few tips to help you navigate the complex web of international relations’
 - (The typo was part of the generation.)

To see more examples, see the codebase.

3. Code

All code for this project is at <https://github.com/Lovkush-A/apart-goodfire>. The readme file explains the codebase, but it is fairly straightforward.

4. Discussion and Conclusion

Despite not getting convincing results in this trial, I am still confident we can recover to a large extent the SAE. The reason is that steering of LLMs is simple - you just add a fixed vector into the activations - so it should be possible to reverse engineer what vector was added in order to produce the results.

Here are possible ways of moving forwards:

- Instead of testing on Goodfire's API, test on open source SAEs. This will include smaller models, and so faster iteration.
- Introduce unit tests / other means of ensuring the code is not buggy. There is a high chance the code is not sound...
- Introduce quantitative methods of measuring success and uncertainty in the steering vectors.
- Use a better methodology for creating the steering vector. Naively looking at the difference is not sound. One alternative is to use gradient descent to find the steering vector:
 - Step 1. Pick a feature.
 - Step 2. Create a dataset (p_i, s_i, x_i) where p 's are prompts, s 's are the strength of nudging and the x 's are the generations from Goodfire nudged with the chosen feature with strength s .
 - Step 3. Find steering vector by using gradient descent on the learning task of next token prediction. Instead of learning all the parameters, we simply need to learn the steering vector.