

Analyzing Dataset Bias with Sparse Autoencoder Features

Nick Jiang

Affiliation

Joseph Tey

Affiliation

With

Goodfire AI and Apart Research

Abstract

When training and finetuning large language models (LLMs), using a large, diverse dataset is crucial for lifting general performance and preventing gaps in model knowledge. From an AI safety standpoint, underrepresented samples and classes in datasets leave models susceptible to adversarial attacks and exploits. While most preventative work for model failures has focused on analyzing model internals or reprogramming weights to be more robust, we instead contribute to the upstream task of bias detection in datasets and identify significant biases in spam classification and code vulnerability datasets. We leverage recent breakthroughs with sparse autoencoders by identifying features most predictive of characterizing different classes and observe distribution shifts along these features across classes.

Keywords: AI Observability, Mechanistic Interpretability, Model Reprogramming

1 Introduction

When training and finetuning large language models (LLMs), using a large, diverse dataset is crucial for lifting general performance and preventing gaps in model knowledge. From an AI safety standpoint, underrepresented samples and classes in datasets leave models susceptible to adversarial attacks and exploits. While most preventative work for model failures has focused on analyzing model internals or reprogramming weights to be more robust, we instead contribute to the upstream task of bias detection in datasets and identify significant biases in spam classification and code vulnerability datasets. We leverage recent breakthroughs with sparse autoencoders by identifying features most predictive of characterizing different classes and observe distribution shifts along these features across classes.

2 Overview

Given a dataset with two classes C_1 and C_2 , our task is to evaluate spurious correlations that exist between data points belonging to C_1 and C_2 . As features may be dead or inaccurately labeled from automated interpretability, we first filter the set of all features into a handful of promising examples that can unveil bias between the two classes. Then, we qualitatively look for and quantitatively measure distribution changes along these filtered features.

Datasets. We choose two binary classification datasets to analyze, the Enron Spam Dataset and Cybernative.ai Code Vulnerability and Security Dataset.

Filtering our feature set. We first identify an initial set of features most casually linked to steering from C_1 to C_2 . Then, we try two different interpretable classifiers, decision trees and logistic regression, to continue pinpointing a minimum set of features that have the highest classification scores. Empirically, we find that both methods can identify features associated with spurious correlations. However, we prefer logistic regression because it allows us to evaluate the effect of additional features without the need to exhaustively test combinations (e.g., pairs, triples, etc.) as decision trees often require for optimal feature selection.

Identifying spurious correlations. We plot the distributions of activations along a feature for each class and measure differences with KL divergence and L2. Our results in 1 not only affirm that predictive features have distribution shifts across different classes but also demonstrate the practicality of using features to identify spurious correlations in a classification dataset.

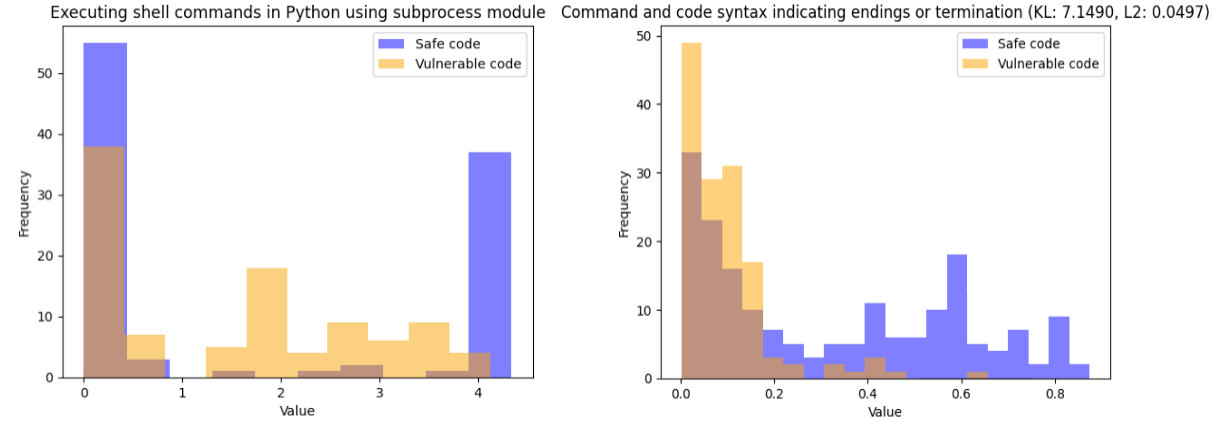


Figure 1: Distributions of the most predictive features from the code vulnerability dataset. These distributions come from collecting the feature activations across 100 randomly selected samples from each class.

3 Code

Key methods. We primarily use the contrastive search endpoint for collecting an initial set of 50 features most promising for steering from C_1 to C_2 . We also make use of the "inspect" endpoint in order to compute the activations of our filtered set of features across 1000 randomly chosen samples from our dataset. We implement our logistic regression and decision tree models using the sklearn library and compute KL divergence using scipy. We included L2 as a metric for comparing distributions because it is simpler and more intuitive to understand than KL divergence. We also explored applying dimensionality reduction techniques like PCA to measure the spread of our filtered features, but we found raw distributions per feature to be the clearest indication of spurious correlations. One caveat with logistic regression is that it is unable to model complex correlational relationships between features and predicted labels (ie. if feature1 and feature2 fire, then the prediction is a "dog") whereas decision trees can due to their nonlinear structure.

Codebase structure. Our codebase is essentially a single notebook marked with sections for extracting features from contrastive search, running a decision tree vs. logistic regression, and performing a feature-level analysis on the original dataset. We also built an interactive interface near the end that highlights tokens based on their feature activations, and we use it to verify that identified features do show up in one class significantly more than the other. Our code can be found here: <https://github.com/josephitey/goodfire-hackathon/>.

4 Discussion and Conclusion

Qualitative results in 1 demonstrate that our method is able to extract features tied to bias in classification datasets. Models trained or fine-tuned on this code vulnerability dataset may learn, for example, to mark code with null terminations as automatically safe or vice versa. Such models may be unable to generalize their knowledge to real-world conditions and may be susceptible to adversarial attacks as a result. While not all of our most predictive features necessarily showed

a clear distribution shift, we believe that this is in part due to our classifiers being unable to pick up the best features. Empirically, we were not able to fool the 8B model with simple prompt tweaks. Given more time, we would also change out the initial constrastive search to an attribution method that more casually ties a predicted output with activated features. Attribution would also allow us to study how models specifically perceive and think about classifying dataset examples, likely strengthening our ability to find real adversarial examples that successfully fool models. Our results indicate the potential for using SAE features for identifying dataset biases, a crucial yet under-explored area within AI safety.

References