Representation Engineering via Control Vectors

Hayden Outlaw
Tulane University / New Orleans, LA
houtlaw@tulane.edu

Joe Wagner
Tulane University / New Orleans, LA
jwagner3@tulane.edu

1 Problem Overview

Our project demonstrates a powerful new form of representation engineering called control vectors. Given a language model, and a dichotomy of predefined traits or behaviors (honest vs dishonest, drunk vs sober, helpful vs unhelpful, verbose vs brief, etc.,), we load pre-generated control vectors, and use them to steer the model's output to include the representation of whatever those control vectors encode. The vectors are learned ahead of time by running contrastive prompts through the model, and then the representations of the target concept are extracted, generally using principal component analysis. Once cached, they can be added to the activation function between layers of the model to control the model's output, and can be further scaled by constants or added jointly as a sum to add multiple representations simultaneously.

2 Data

While our project is centered around pre-trained foundation models, we still require external data and code. Specifically we lean on two pre-developed libraries: the *RepEng* (Vogel, 2024) library by Theia Vogel, and the *RepE* (Zou et al., 2023) library, which both provide frameworks for generating and evaluating these control vectors on open source models. Most critically, the *RepE* model contains a list of manually created objectively true and false statements, which are required for the generation of control vectors, which we will include for our own usage instead of identifying potential options from scratch.

We also use open-source foundation models; since we require the ability to manually configure and edit the activation functions between layers, we require a model that is as basic and accessible as possible. For this, we utilize the Mistral *Mistral-7B-Instruct-v0.1* model (Jiang et al., 2023), which is advanced enough to allow for the different behaviors we aim to isolate while still being more accessible than other commercially developed frameworks. While we have access to pre-generated supplemental data required to train these specific mixture of expert models, if we were to implement this with other models such as transformers, we potentially would have to either edit or recreate them ourselves.

3 Methods

Similar to the concept of 'memory' cells or attention heads in transformer-based frameworks, control vectors aim to induce behavior by adding representations of concepts to layers before taking their activation function. However, this method differs in the order of operations: instead of trying to learn representations of features ahead of time, we aim to learn as accurate of a representation for a concept one single time, and then once that representation is captured, enforce behavior of this representation in future evaluations of the model without re-learning them. This allows for more overt control of the model's behavior, and is more robust and effective than additional methods for controlling model output such as prompt engineering.

Furthermore, since these vectors are representations of concepts internally within the model, they can be imposed as a linear combination for an additional degree of freedom in selecting model behavior to enforce. While theoretically an infinite number of unique control vectors could be added, at some point they have to not overpower the learned weights within the model - so when combined they are often scaled or regularized to a greater degree.

4 Experiments and Results

So far, we have successfully been able to deploy the *RepEng* and *RepE* libraries, and use them in generating control vectors and modified responses. We utilize the Mistral-7B-Instruct-v0.1 model from HuggingFace, as it's what was originally used in *Zou et al*(Zou et al., 2023), although we might use the 0.2 version that has since been launched.

We have a better understanding of the actual mechanism by which the dataset is generated, specifically regarding the the "[/INST]" tokens within Mistral models and how they allow users to pass instructions directly into the prompts without using more complicated prompt engineering techniques. Mixture of Experts models, which were the standard used in relevant research, are significantly different than the more familiar transformer models, as they contain learned gating and expert weights - however between each layer the mechanism of an activation function remains relatively similar, and so we can sidestep investigating the more literal learning mechanism for these new models for the most part.

Overall, while the actual computation is perhaps complicated, in principle this is a very powerful concept that is relatively simple compared to the notion of attention or other additions to activation functions between layers.

5 Related Work

1. *Representation Engineering: A Top-Down Approach to AI Transparency* (Zou et al., 2023)

   Zou et al's paper on representation engineering formalizes current progress in the field by prioritizing representations of higher-level concepts above neurons or activation functions for framing models. They introduce the concept of control vectors in order to extract these high-level representations of networks. The utility of control vectors is demonstrated for model analysis and AI transparency. They create control vectors using contrastive prompting and primcipal component analysis, which are then added onto the activation functions to enforce behaviors, similar to our work. Whereas they focus on model interpretability

2. *Towards Monosemanticity: Decomposing Language Models with Dictionary Learning* (Bricken et al., 2023)

   Bricken et al., examine language model interpretability through the lens of monosemanticity and polysemanticity - the study of how one individual neuron or unit can encode information for one or multiple embeddings as a component of a combination of other neurons. They use a sparse autoencoder to generate learned features, as opposed to principal component analysis, which is a more involved process that leans on a greater amount of prior work - however, they had relative success in using these models to extract a high proportion of monosemantic features, and in a way that was relatively model architecture agnostic.

3. *Activation addition: Steering language models without Optimization* (Turner et al., 2023)

   Turner et al do not lean on the concept of 'representation' as much, but do propose the framework of modifying activation functions to steer or control model outputs, specifically by using contrasting prompt pairs to learn a vector, which is then injected into the activation function when multiplied by some parameter scalar. This paper predominantly focuses on sentiment steering and improving model performance, but is much less concerned with safety, adversarial querying, and jailbreaking than Zou et al.,

4. *A Tutorial on Network Embeddings* (Haochen Chen, 2018)

   Chen et al offer an overview of different ways to extract representation embeddings from models beyond PCA. This is different than the high-level concepts we are trying to learn with control vectors, and rather focuses on the applications of low-dimensional latent node representation. Some of the methods covered include isomaps and local linear embeddings. While somewhat out of the scope of our project, the unsupervised methods are potential strategies we will explore to improve our model.

5. *Representation Engineering Mistral-7B an Acid Trip* (Vogel, 2024)

   Extends and explains the functionality of the repeng python library, which allows for easy use of control vectors with pytorch. Looks beyond the high level representation side of RepE and instead compares it to current methods including prompt engineering and fine tuning. Packaged code provided allows for generating control vectors from contrastive prompts. Whereas Vogel focuses on a wide range of applications for control vectors, our project instead hones in on giving users a variety of trained control vectors which all elicit unique responses.

6 Division of Labor

Between our two backgrounds, for the time being we have decided to begin construction of the web utility and the backend code separately, and then work to join them in the middle. Since a lot of our project involves generating and caching data ahead of time for the demo, once we have a functional framework with one or two attributes that you can select, we are essentially limited only by the amount of features we want to add and the amount of time we have to do the corresponding model inference. Once we have a functioning demo with one or two behaviors that users can select and query the model with, since most of the labor in computing the control vectors is done ahead of time and cached we can add as many control behaviors as we can devleop in the time allotted.

7 Timeline

Up to this point, since we have successfully compiled the library code and configured the starting model, there are a few directions in which we could proceed. We could work on expanding the different options for which representations are essentially cached for the demo for a wider variety of behaviors for the demo, or we could add support for visualization of some kind of the representation embedding of whatever behavior is selected. We could also add simultaneous generation of the responses of the contrasting vectors, or add UI support in our demo for the creation of novel control vectors based on inputs.

References

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, page 2.

Rami Al-Rfou Steven Skiena Haochen Chen, Bryan Perozzi. 2018. A tutorial on network embeddings. *arXiv preprint arXiv:1808.02590*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Alex Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*.

Theia Vogel. 2024. repeng.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.