What Cosine Similarity of Label Representations Can and Cannot Tell us

Beatrix M. G. Nielsen [1] **Andreas Grivas** [2]

Abstract

Cosine similarity is often used to measure the similarity of vector representations of neural network models. However, the cosine similarity of representations is not guaranteed to tell us anything about model probabilities. In this paper we show that for a softmax classifier, be it an image classifier or an autoregressive language model, the cosine similarity between label representations (called unembeddings in the paper) does not give any information on the probabilities assigned by the model. Specifically, we prove that given two unembeddings, it is possible to create another model which assigns the same probabilities for all inputs, but where the cosine similarity between the representations is now either 1 or -1. We also show that for a sigmoid classifier (where each input can be assigned multiple labels), all pairwise cosine similarities between the unembeddings define the set of possible label combinations. However, for softmax classifiers (where each input is assigned a ranking of the labels from most to least likely), we need all pairwise cosine similarities between all *differences* of unembeddings to know which rankings the model can predict. We conclude that it is misleading to interpret the cosine similarity between unembeddings without reference to the classifier that produced them.

1. Introduction

Suppose we have a language model (of the model class we introduce below Equation (1)) and we consider the vector representations for the two tokens "dog" and "cat". If we measure the cosine similarity between the two to be 0.95, it is tempting to say that this is because "dog" and "cat" are tokens which co-occur in many similar contexts (reminiscent of Latent Semantic Analysis (Deerwester et al., 1990) and Word2Vec (Mikolov et al., 2013)), so the model will assign similar probabilities to these two tokens. However, this reasoning is flawed for softmax classifiers: it is possible to make a second model which matches the token probabilities of the original model for all inputs, but where the cosine similarity between "dog" and "cat" is now -1. It is also possible to make a model which assigns very different probabilities to the "dog" and "cat" tokens, but where the cosine similarity of their representations is 1. This means that the cosine similarity between representations of tokens is disconnected from assigned probabilities and therefore should not be used for explaining those probabilities.

While some studies that use cosine similarity on token representations are aware of its limitations, *e.g.* Land & Bartolo (2024, Appendix A), and Steck et al. (2024) shows that cosine similarity can be arbitrary for certain linear models, there is to our knowledge no reference in the literature that highlights the problem for general softmax classifiers. We fill this gap and further highlight that it is misleading to interpret the cosine similarity between unembeddings without reference to the classifier that produced them. To show this, we characterise when the decision regions of softmax and sigmoid classifiers are determined by a matrix of cosine similarities between their unembeddings. In particular, if the unembeddings arise from a sigmoid classifier, all pairwise cosine similarities of the unembeddings determine its decision regions. In contrast, for the unembeddings of a softmax classifier, we need all pairwise cosine similarities between all *differences* of unembeddings to determine its fine-grained decision regions; pairwise cosine similarities of unembeddings are not enough.

Our contributions are as follows:

- For a softmax classifier (Equation (1)), we show that since probabilities are invariant to vector translations (Lemma 3.1) but cosine similarity is not (Lemmas 3.2 and 3.3), these two properties of the model are disconnected (Theorem 3.4). Illustration in Figure 1.
- For a sigmoid classifier (Equation (2)), we show that cosine similarities of unembeddings **can** tell us the

[1]IT University of Copenhagen, Denmark [2]School of Mathematics, University of Edinburgh, UK. Correspondence to: Beatrix M. G. Nielsen <beat@itu.dk>, Andreas Grivas <agrivas@ed.ac.uk>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

possible label combinations (Lemma 5.3).

- For a softmax classifier, we show that cosine similarities of unembeddings **cannot** tell us the possible rankings (Theorem 5.5), for this we need cosine similarities between *differences* of unembeddings (Theorem 5.6).

2. The Softmax Classifier

Let $\mathbf{x} \in \mathcal{X}$ be inputs and $\mathcal{Y}$ a set of k labels. We define a softmax classifier as a model using functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^d$, $\mathbf{g} : \mathcal{Y} \to \mathbb{R}^d$ for $d \in \mathbb{Z}_{>0}$, which assigns probabilities, p, to mutually exclusive labels, $y \in \mathcal{Y}$, given inputs, $\mathbf{x} \in \mathcal{X}$, as:

$$p(y \mid \mathbf{x}) = \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y'))} \,. \tag{1}$$

So for each input the model makes a categorical distribution over the labels. The $\mathbf{f}(\mathbf{x})$ are the representations of the inputs, and we call these the embeddings. The $\mathbf{g}(y)$ are the representations of the labels and we call these the unembeddings. In this paper we focus on the unembeddings, as we are interested specifically in what their properties can tell us about model outputs. The model class in Equation (1) is very broad including any image classifier using a softmax on the last logits, whether the encoder is a single Convolutional Neural Network (LeCun et al., 1989), a ResNet (He et al., 2016) or a vision transformer (Dosovitskiy et al., 2020). The model class also includes large language models from early incarnations like BERT (Devlin et al., 2019)[1] and GPT-2 (Radford et al., 2019) to more recent models like GPT-3 (Brown et al., 2020)[2] and gpt-oss-120B (OpenAI, 2025, see released code).

3. Cosine of Unembeddings vs Probabilities

The first result of this paper relies on two properties. First, for a softmax classifier, adding some vector $\mathbf{v}$ to all unembeddings, does not change the probability distributions of the model (this is not new, but we state it in Lemma 3.1 and provide a full proof in Section A for completeness).

Lemma 3.1. *Let $\mathbf{v} \in \mathbb{R}^d$. For a softmax classifier as in Equation (1), adding $\mathbf{v}$ to all unembeddings, does not change the probability $p(y \mid \mathbf{x})$.*

Second, adding such a vector to the unembeddings can greatly change the cosine similarities. In the next two lemmas we see that for two vectors, we can always find a translation such that the translated vectors have cosine similarity -1 (Lemma 3.2) or 1 (Lemma 3.3). Proofs in Section B.

Lemma 3.2. *Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ be vectors with $\mathbf{a} \neq \mathbf{b}$. We can always choose $\mathbf{v} \in \mathbb{R}^d$ such that $\cos(\mathbf{a} + \mathbf{v}, \mathbf{b} + \mathbf{v}) = -1$.*

[1] See Roeder et al. (2021, Appendix D.3).

[2] See Radford et al. (2018, equation 2)

Lemma 3.3. *Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ be two vectors. We can always choose $\mathbf{v} \in \mathbb{R}^d$ such that $\cos(\mathbf{a} + \mathbf{v}, \mathbf{b} + \mathbf{v}) = 1$.*

3.1. Equivalent Models can have very Different Cosine Similarities Between Unembeddings

Theorem 3.4. *Let $(\mathbf{f}, \mathbf{g})$ be a softmax classifier as in Equation (1). For two labels $y_i, y_j \in \mathcal{Y}$, assume the unembeddings of the labels are not equal, $\mathbf{g}(y_i) \neq \mathbf{g}(y_j)$. Let $\cos(\mathbf{g}(y_i), \mathbf{g}(y_j)) = c$ for some $c \in [-1, 1]$. We can then construct two equivalent models (all assigned probabilities will be the same for all inputs), $(\mathbf{f}', \mathbf{g}')$ and $(\mathbf{f}'', \mathbf{g}'')$, where for $(\mathbf{f}', \mathbf{g}')$ we have $\cos(\mathbf{g}'(y_i), \mathbf{g}'(y_j)) = -1$ and for $(\mathbf{f}'', \mathbf{g}'')$, we have $\cos(\mathbf{g}''(y_i), \mathbf{g}''(y_j)) = 1$.*

Proof. By Lemma 3.1, we can make an equivalent model by adding the same vector, $\mathbf{v}$ to all unembeddings. For constructing the model $(\mathbf{f}', \mathbf{g}')$, we choose $\mathbf{v}$ as in Lemma 3.2 and for $(\mathbf{f}'', \mathbf{g}'')$ we choose $\mathbf{v}$ as in Lemma 3.3. □

Theorem 3.4 says that the cosine similarity between two unembeddings does not tell us anything about the probabilities which the model assigns to the corresponding labels. For an example of three models which for all inputs assign the same probabilities, but where cosine similarity between their unembeddings vary greatly see Figure 1 and details in the Appendix, Section C.

With respect to other distance measures, Theorem 3.4 means that the normalized Euclidean distance between unembeddings will also not tell us anything about the probabilities. The theorem also says that the dot product between unembeddings can have different signs for two equivalent models, and therefore the dot-product will also not give us information on the probabilities[3].

Lemmas 3.2 and 3.3 rely on translations of the unembeddings, therefore one might hope that centering the unembeddings (*i.e.* making the mean across each feature dimension zero) could help. However, as we show in Sections C.2 and C.3 in the Appendix, this still does not guarantee that high cosine similarity leads to high probabilities for the same inputs and fixing the length of the unembeddings is also not enough to get a guarantee.

4. The Sigmoid Classifier

Let $\mathbf{x} \in \mathcal{X}$ be inputs and $\mathcal{V}$ a set of k labels. In multi-label classification, our goal is to classify an input $\mathbf{x}$ by assigning it a subset of non-mutually exclusive labels, $\mathbf{v} \subseteq \mathcal{V}$. We

[3] Just to complete the list of commonly used distances: The Euclidean distance will also not give us information about the probabilities, since if we scale all unembeddings by $c > 0$ and all embeddings by $1/c$, then we do not change the probabilities, but we do change Euclidean distances.

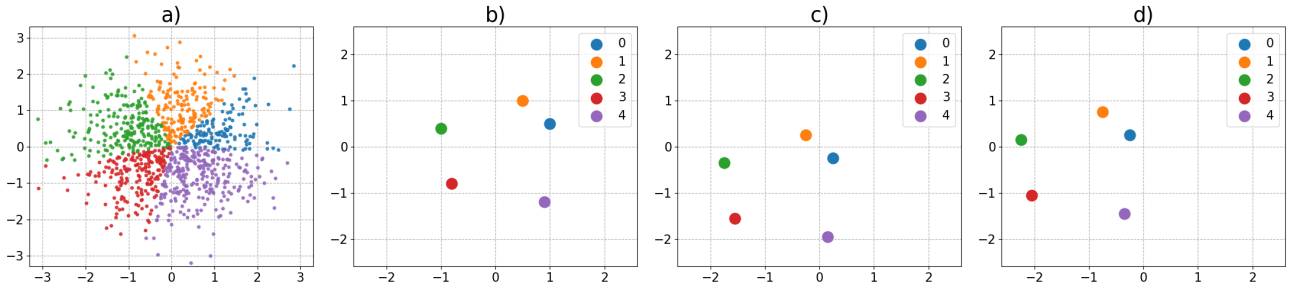

Figure 1. Example of three models which give same probabilities to all labels for all inputs, but where cosine similarities differ between the unembeddings, see the Appendix Section C for details. **a)** Embeddings coloured by highest probability label. These are the same for all three models. **b)** Unembeddings for model 1. Cosine between labels 0 and 1 is about 0.8. **c)** Unembeddings for model 2. Cosine between labels 0 and 1 is -1. **d)** Unembeddings for model 3. Cosine between labels 0 and 1 is 1.

model such label assignments via indicator variables of the form $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}$ where $\mathcal{Y} = \{+1, -1\}^k$, and $y_v = +1$ denotes that label v was assigned to $\mathbf{x}$ while $y_v = -1$ denotes that it was not. We define a sigmoid classifier using functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^d$, $\mathbf{g} : \mathcal{V} \to \mathbb{R}^d$ for some $d \in \mathbb{Z}_{>0}$, which assigns probabilities to each label $v \in \mathcal{V}$ given inputs, $\mathbf{x} \in \mathcal{X}$ as:

$$p(y_v \mid \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{f}(\mathbf{x})^\top \mathbf{g}(v))}. \tag{2}$$

So for each input the model makes a Bernoulli distribution for each label. The well known Word2Vec (Mikolov et al., 2013) model using skip-gram with negative sampling can be interpreted as a sigmoid classifier (see Section D.1 for details), where the goal is to classify which tokens are likely to co-occur together in a token window. Therefore for this model, the intuition that high cosine similarity connects to co-occurrence makes more sense.

5. Cosine of Unembeddings vs Decision Regions

5.1. Cosine Similarities of Unembeddings Determine Sigmoid Decision Regions

We let $\mathbf{W} \in \mathbb{R}^{k \times d}$ be the unembedding matrix with i'th row $\mathbf{g}(y_i)^\top$, and use $\mathbf{w}_i$ as a shorthand. The decision region for a label combination $(y_1, y_2, \ldots, y_k) \in \{+1, -1\}^k$ is given by $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^d$:

$$y_i \mathbf{w}_i^\top \mathbf{f}(\mathbf{x}) > 0, \quad \forall i : 1 \leq i \leq k, \tag{3}$$

see Section D.2 for details. Importantly, when $d < k$, there are label combinations that cannot be predicted (Cover, 1965; Grivas et al., 2024).

Definition 5.1 (Sign Pattern)**.** We determine the set of feasible label combinations as the sign pattern of $\mathbf{W} \in \mathbb{R}^{k \times d}$:

$$\mathcal{S}(\mathbf{W}) = \left\{ \operatorname{sign}(\mathbf{W}\mathbf{z}) : \mathbf{z} \in \mathbb{R}^d,\ \mathbf{w}_i^\top \mathbf{z} \neq 0,\ 1 \leq i \leq k \right\}, \tag{4}$$

where we disallow zeroes because we want to focus on the predicted label combinations and not parts of the decision boundaries[4]. The $\operatorname{sign}()$ function is applied element-wise.

Usually the $\mathbf{z}$ in Equation (4) would be our embeddings, $\mathbf{f}(\mathbf{x})$. However, if $\mathbf{f}$ is not surjective, and some decision regions of the classifier are not reached, we would get further restrictions which we will not go into here. When $d \geq k$ and $\operatorname{rank}(\mathbf{W}) \geq k$ all label combinations can be predicted, *i.e.* $|\mathcal{S}(\mathbf{W})| = 2^k$, and the sign pattern is not useful for telling such unembeddings apart. However, when $d < k$, there must exist label combinations that cannot be predicted and therefore the sign pattern can be used to distinguish between models. To focus on this more interesting case, imagine in the following that $d < k$ and $\operatorname{rank}(\mathbf{W}) \leq d$. **Can we determine which label combinations are feasible from the cosine similarities of the unembeddings?** We show that we can: we introduce the cosine similarity matrix $\mathbf{C}$ which we will show determines the sign pattern.

Definition 5.2 (Cosine Similarity Matrix)**.** Let $\mathbf{C} = \mathbf{C}(\mathbf{W}) \in \mathbb{R}^{k \times k}$ be the symmetric matrix with entries $C_{i,j} = \cos(\mathbf{w}_i, \mathbf{w}_j)$, $i, j \in \{1, \ldots, k\}$, where:

$$\cos(\mathbf{w}_i, \mathbf{w}_j) = \frac{\mathbf{w}_i^\top \mathbf{w}_j}{\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2}. \tag{5}$$

Lemma 5.3. *Let $\mathbf{W} \in \mathbb{R}^{k \times d}$ with non-zero rows. Then the cosine similarity matrix $\mathbf{C}(\mathbf{W})$ of the unembedding rows determines $\mathcal{S}(\mathbf{W})$.*

Proof in Section E. This means that given the unembeddings and all cosine similarities between them, we know which combinations of labels can be predicted by our model.

[4] Allowing logits to be zero would complicate our set notation: our set would also count decision boundaries, when we are only interested in the set of outputs that can be predicted. Moreover, logits that are exactly zero are unlikely to occur, so ignoring the zero case is not an omission that matters in practice.

5.2. Cosine Similarities of Differences of Unembeddings Determine Softmax Decision Regions

For the softmax case, the decision region for label t is given by feature vectors $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^d$ that satisfy:

$$(\mathbf{w}_t - \mathbf{w}_i)^\top \mathbf{f}(\mathbf{x}) > 0, \quad \forall i : 1 \leq i \leq k, \quad i \neq t, \tag{6}$$

see Section F.1 for details. We note that in contrast to the sigmoid case, we now take pairwise differences of the rows of $\mathbf{W}$. We define the softmax decision regions by introducing the braid matrix (Stanley, 2004, lecture 1), $\mathbf{B} \in \mathbb{R}^{\binom{k}{2} \times k}$. This is the matrix whose rows are differences of unit vectors $\mathbf{e}_i^\top - \mathbf{e}_j^\top$, $1 \leq i < j \leq k$ in lexicographic order, see Section F for more details. Note that $\mathbf{W}' = \mathbf{BW}$ contains all pairwise differences of the rows of $\mathbf{W}$.

Definition 5.4 (Ranking Pattern)**.** We define the set of feasible rankings for a classifier $\mathbf{W}$ as the ranking pattern:

$$\mathcal{R}(\mathbf{W}) = \left\{ \begin{array}{ll} \operatorname{sign}(\mathbf{BWz}) : & \mathbf{z} \in \mathbb{R}^d, \\ & (\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{z} \neq 0, \\ & 1 \leq i < j \leq k \end{array} \right\}, \tag{7}$$

where as in the sign pattern case, we do not consider any points that fall on the decision boundary.

The ranking pattern is important as it determines which rankings of labels our softmax classifier is able to produce or not. If $\operatorname{rank}\left(\widetilde{\mathbf{W}}\right) \geq k-1$, where $\widetilde{\mathbf{W}}$ is the centered $\mathbf{W}$ with column means subtracted, then the model will be able to assign any ranking of the labels, *i.e.* $|\mathcal{R}(\mathbf{W})| = k!$. However, when $\operatorname{rank}\left(\widetilde{\mathbf{W}}\right) < k-1$, *e.g.* when $d < k-1$, many rankings will be infeasible and therefore missing from the set (Cover, 1967; Grivas et al., 2022). **Can we determine which rankings are feasible from the matrix of cosine similarities?**

As we prove via counter-example in Section G, this is not possible: while $\mathbf{C}(\mathbf{W})$ determines $\mathcal{S}(\mathbf{W})$, it does not determine $\mathcal{R}(\mathbf{W})$, and therefore neither does $\mathcal{S}(\mathbf{W})$, via Lemma 5.3. On the other hand, the cosine similarities of all the differences of unembeddings do determine the ranking pattern, see the proof in Section F.

Theorem 5.5. $\mathbf{C}(\mathbf{W})$ *(and by extension, $\mathcal{S}(\mathbf{W})$) does not determine $\mathcal{R}(\mathbf{W})$.*

Theorem 5.6. $\mathbf{C}(\mathbf{BW})$ *determines $\mathcal{R}(\mathbf{W})$.*

Verification of Results While it is intractable to check the sign pattern and ranking pattern of large matrices due to the combinatorial explosion of outputs, we numerically checked Lemma 5.3 and Theorem 5.6 for unembedding matrices $\mathbf{W}$ with $n \in \{3, 5, 10\}$ and $d \in \{2, \ldots \min(6, n-1)\}$. We also provide a check for the counter-example of Theorem 5.5. Our code implementing the checks is available on github.[5]

6. Discussion and Further Questions

Our results highlight that the way we use cosine similarity cannot be independent of the model. For sigmoid classifiers, we can use the pairwise cosine similarity between all unembeddings to get which label combinations the model can predict (Lemma 5.3). On the other hand, for a softmax classifier we need the pairwise cosine similarity between all *differences* of unembeddings to determine which rankings the model can predict (Theorem 5.6). We highlight that we only find which label combinations/rankings are possible, we do not know whether they will be assigned to any input, for this we need to also consider the embeddings, $\mathbf{f}(\mathbf{x})$.

Our result in Theorem 3.4 suggests that when working with softmax classifiers, we should not use cosine similarity between the unembeddings to say something about the assigned probabilities of the models. On the other hand, this result says nothing about whether cosine similarity is a meaningful measure to use between embeddings, as for example the probabilities of a softmax classifier are not invariant to vector translations of the input representations.

Nielsen et al. (2025) saw that when comparing token representations with each other it often resulted in nuisance hubness and therefore advocate for hubness reduction techniques. However, this work shows that the similarity measures themselves do not carry information about the probabilities, and this will probably not be fixed by doing hubness reduction. Related to this: In high dimensions for random vectors, cosine distance will often concentrate around 1. So if for trained models the cosine similarities vary a lot, this means that the unembeddings of trained models behave very differently from random matrices. What this "different from random behaviour" means is an interesting question which we hope will inspire future work.

The fact that cosine similarity of unembeddings are not directly connected to probabilities, might also be part of the explanation for why the BERT [CLS] token is suboptimal for retrieval using cosine similarity, see *e.g.* Reimers & Gurevych (2019, Table 1).

In Section 4 of (Lee et al., 2025), they show that some models have quite high correlations between their pairwise (cosine) distance matrices. But if cosine similarity between token unembeddings is not about probabilities, what do these high correlations tell us? Do they mean that there is a bias in the models for learning specific solutions? And if yes, where does such a bias come from?

[5] https://github.com/bemigini/how-not-to-use-cosine-sim

Related to this work, (Steck et al., 2024) consider cosine similarity between representations of two types of linear models. They identify several cases where the cosine between representations relies on an arbitrary matrix and therefore cannot give us information about the model. Note that in our definitions of classifiers, the (un)embedding functions, $\mathbf{f}, \mathbf{g}$, can be highly non-linear.

As we mentioned in Section 4, Word2Vec can be interpreted as a sigmoid classifier (Section D.1) and therefore pairwise cosine similarities between unembeddings will for this model give us the possible label combinations. It is possible that when the field of natural language processing transitioned to mainly using softmax classifiers instead of sigmoid classifiers like Word2Vec, some of the intuition about how cosine similarity could be used might have carried over. We hope that with this paper we can make the field more aware of which assumptions need to be satisfied for cosine similarity between unembeddings to be useful.

7. Conclusion

This paper shows examples of what cosine similarity can and cannot identify. First, for a softmax classifier high (or low) cosine similarity between unembeddings **cannot** be used to say that probabilities for the corresponding labels will be high (or low) at the same time. Second, for a sigmoid classifier all pairwise cosine similarities between unembeddings **can** identify which combinations of labels can be predicted. Third, for a softmax classifier, the pairwise cosine similarities **cannot** identify which rankings can be predicted, but the pairwise cosine similarities between all *differences* of unembeddings **can**. Thus, before using cosine similarity one should consider whether it can answer the question of interest and which kind of classifier is relevant.

Impact Statement

This paper attempts to better understand how cosine similarity can be used to say something about machine learning models based on either a sigmoid or a softmax classifier. It is difficult to say exactly which consequences our work will have on society, but we hope an improved understanding will lead to better and safer models.

Acknowledgments

B. M. G. N. was supported by the Novo Nordisk Foundation grant NNF24OC0092612. A.G. was supported by ERC grant ("Numerical Analysis for Stable AI", 101198795).

References

Björner, A., Las Vergnas, M., Sturmfels, B., White, N., and Ziegler, G. M. *Oriented Matroids*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1999. doi: 10.1017/CBO9780511586507.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Cover, T. M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electron. Comput.*, 14:326–334, 1965.

Cover, T. M. The number of linearly inducible orderings of points in d-space. *SIAM Journal on Applied Mathematics*, 15(2):434–439, 1967. ISSN 00361399. URL http://www.jstor.org/stable/2946294.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.*, 41(6):391–407, 1990. doi: 10.1002/(SICI)1097-4571(199009)41:6\⟨391::AID-ASI1\⟩3.0.CO;2-9.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Grivas, A., Bogoychev, N., and Lopez, A. Low-rank softmax can have unargmaxable classes in theory but rarely in practice. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6738–6758, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.465. URL https://aclanthology.org/2022.acl-long.465/.

Grivas, A., Vergari, A., and Lopez, A. Taming the sigmoid bottleneck: Provably argmaxable sparse multi-label classification. In Wooldridge, M. J., Dy, J. G., and Natarajan, S. (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on*

Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, pp. 12208–12216. AAAI Press, 2024. doi: 10.1609/AAAI.V38I11.29110. URL https://doi.org/10.1609/aaai.v38i11.29110.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Land, S. and Bartolo, M. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11631–11646, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.649. URL https://aclanthology.org/2024.emnlp-main.649/.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Lee, A., Weber, M., Viégas, F., and Wattenberg, M. Shared global and local geometry of language model embeddings. *arXiv preprint arXiv:2503.21073*, 2025.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Nielsen, B. M. G., Macocco, I., and Baroni, M. Prediction hubs are context-informed frequent tokens in llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23715–23745, 2025.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162/.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Roeder, G., Metz, L., and Kingma, D. On linear identifiability of learned representations. In *International Conference on Machine Learning*, pp. 9030–9039. PMLR, 2021.

Stanley, R. P. An introduction to hyperplane arrangements. In *Lecture notes, IAS/Park City Mathematics Institute*, 2004.

Steck, H., Ekanadham, C., and Kallus, N. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, pp. 887–890, 2024.

A. Proof that Adding Vector does not Change Probabilities

We recall Lemma 3.1:

Lemma 3.1. *Let* $\mathbf{v} \in \mathbb{R}^d$*. For a softmax classifier as in Equation* (1)*, adding* $\mathbf{v}$ *to all unembeddings, does not change the probability* $p(y \mid \mathbf{x})$*.*

Proof. Let $v \in \mathbb{R}^d$. We will consider the calculation of probabilities for a model which has $\mathbf{g}(y) + \mathbf{v}$ as unembeddings for all $y \in \mathcal{Y}$ and show that this is the same as for the original model. Let $p'(y \mid \mathbf{x})$ denote the probability assigned by the model with $\mathbf{g}(y) + \mathbf{v}$ as unembeddings. We then have that

$$
\begin{aligned}
p'(y \mid \mathbf{x}) &= \frac{\exp(\mathbf{f}(\mathbf{x})^\top(\mathbf{g}(y) + \mathbf{v}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{f}(\mathbf{x})^\top(\mathbf{g}(y') + \mathbf{v}))} && (8)\\
&= \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y) + \mathbf{f}(\mathbf{x})^\top \mathbf{v})}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y') + \mathbf{f}(\mathbf{x})^\top \mathbf{v})} && (9)\\
&= \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y)) \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{v})}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y')) \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{v})} && (10)\\
&= \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y'))} && (11)\\
&= p(y \mid \mathbf{x}). && (12)
\end{aligned}
$$

□

B. Proofs that Adding Vector can Change Cosine

B.1. Full Proof that Adding Vector can make Cosine -1

We recall Lemma 3.2:

Lemma 3.2. *Let* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ *be vectors with* $\mathbf{a} \neq \mathbf{b}$*. We can always choose* $\mathbf{v} \in \mathbb{R}^d$ *such that* $\cos(\mathbf{a} + \mathbf{v}, \mathbf{b} + \mathbf{v}) = -1$*.*

Proof. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ be two vectors such that $\mathbf{a} \neq \mathbf{b}$. We first note that for any two vectors, $\mathbf{u}, \mathbf{w} \in \mathbb{R}^d$, if $\mathbf{u} + \mathbf{w} = 0$, then $\cos(\mathbf{u}, \mathbf{w}) = -1$. This comes from the following:

$$
\begin{aligned}
\mathbf{u} + \mathbf{w} = 0 &\implies \mathbf{u} = -\mathbf{w} && (13)\\
&\implies \cos(\mathbf{u}, \mathbf{w}) = \cos(-\mathbf{w}, \mathbf{w}) = -1\ .
\end{aligned}
$$

We therefore choose $\mathbf{v} = -(\mathbf{a} + \frac{1}{2}(\mathbf{b} - \mathbf{a}))$ and see that

$$
\begin{aligned}
(\mathbf{a} + \mathbf{v}) + (\mathbf{b} + \mathbf{v}) &= \mathbf{a} - \mathbf{a} - \frac{1}{2}(\mathbf{b} - \mathbf{a}) + \mathbf{b} - \mathbf{a} - \frac{1}{2}(\mathbf{b} - \mathbf{a}) && (14)\\
&= \mathbf{b} - \mathbf{a} - (\mathbf{b} - \mathbf{a}) = 0\ . && (15)
\end{aligned}
$$

This means that $\cos(\mathbf{a} + \mathbf{v}, \mathbf{b} + \mathbf{v}) = -1$ by Equation (13). □

B.2. Full Proof that Adding Vector can make Cosine 1

We recall Lemma 3.3:

Lemma 3.3. *Let* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ *be two vectors. We can always choose* $\mathbf{v} \in \mathbb{R}^d$ *such that* $\cos(\mathbf{a} + \mathbf{v}, \mathbf{b} + \mathbf{v}) = 1$*.*

Proof. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ be two vectors. If $\mathbf{a} = \mathbf{b}$, $\cos(\mathbf{a}, \mathbf{b}) = 1$ and we are done. Therefore assume $\mathbf{a} \neq \mathbf{b}$. In particular we have that $\mathbf{b} - \mathbf{a}$ is not the zero vector. We now note that for any two vectors, $\mathbf{u}, \mathbf{w} \in \mathbb{R}^d$, if $\mathbf{u} - c\mathbf{w} = 0$ for $c \in \mathbb{R}_{>0}$, then

$\cos(\mathbf{u}, \mathbf{w}) = 1$. This comes from the following:

$$\begin{aligned} \mathbf{u} - c\mathbf{w} = 0 &\implies \mathbf{u} = c\mathbf{w} \\ &\implies \cos(\mathbf{u}, \mathbf{w}) = \cos(c\mathbf{w}, \mathbf{w}) \\ &= \frac{c\mathbf{w} \cdot \mathbf{w}}{\|c\mathbf{w}\|\|\mathbf{w}\|} = \frac{c\|\mathbf{w}\|^2}{c\|\mathbf{w}\|\|\mathbf{w}\|} = 1\,. \end{aligned} \tag{16}$$

We therefore choose $\mathbf{v} = -\mathbf{a} + \frac{1}{2}(\mathbf{b} - \mathbf{a})$ and see that

$$\mathbf{a} + \mathbf{v} = \mathbf{a} - \mathbf{a} + \frac{1}{2}(\mathbf{b} - \mathbf{a}) = \frac{1}{2}(\mathbf{b} - \mathbf{a}) \tag{17}$$

and

$$\mathbf{b} + \mathbf{v} = \mathbf{b} - \mathbf{a} + \frac{1}{2}(\mathbf{b} - \mathbf{a}) = \frac{3}{2}(\mathbf{b} - \mathbf{a})\,. \tag{18}$$

So we have that $(\mathbf{b} + \mathbf{v}) = 3(\mathbf{a} + \mathbf{v})$, which means that $\cos(\mathbf{a} + \mathbf{v}, \mathbf{b} + \mathbf{v}) = 1$ by the observation in Equation (16). □

C. Examples with Full Details

Code for the examples and plots can be found on github
https://github.com/bemigini/how-not-to-use-cosine-sim.

C.1. Example using Unrestricted Unembeddings

We assume three models as in Figure 1, where all three models have embeddings as in Figure 1 **a)**, and **b)**, **c)**, **d)** are the three different sets of unembeddings. The unembeddings of the first model corresponding to the five labels are in order 0 to 4:

$$\mathbf{l}_0 = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}\,,\ \mathbf{l}_1 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}\,, \mathbf{l}_2 = \begin{pmatrix} -1 \\ 0.4 \end{pmatrix}\,,\ \mathbf{l}_3 = \begin{pmatrix} -0.8 \\ -0.8 \end{pmatrix}\,,\ \mathbf{l}_4 = \begin{pmatrix} 0.9 \\ -1.2 \end{pmatrix}\,.$$

We see that the cosine similarity between unembeddings corresponding to labels 0 and 1 is:

$$\cos(\mathbf{l}_0, \mathbf{l}_1) = \frac{1}{1.25} = 0.8\,. \tag{19}$$

The unembeddings for the second model (Figure 1 **c)**) are constructed by taking $\mathbf{v}$ as in Lemma 3.2, so we take

$$\begin{aligned} \mathbf{v} &= -\mathbf{l}_0 - \frac{1}{2}(\mathbf{l}_1 - \mathbf{l}_0) \\ &= -\begin{pmatrix} 1 \\ 0.5 \end{pmatrix} - \frac{1}{2}\left(\begin{pmatrix} 0.5 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}\right) = \begin{pmatrix} -0.75 \\ -0.75 \end{pmatrix}\,. \end{aligned} \tag{20}$$

The unembeddings of the second model are then $\mathbf{l}_i + \mathbf{v}$ and the cosine similarity between $\mathbf{l}_0 + \mathbf{v}$ and $\mathbf{l}_1 + \mathbf{v}$ is:

$$\cos(\mathbf{l}_0 + \mathbf{v}, \mathbf{l}_1 + \mathbf{v}) = -1\,. \tag{21}$$

The unembeddings for the third model (Figure 1 **d)**) are constructed by taking $\mathbf{v}$ as in Lemma 3.3. That is,

$$\mathbf{v} = -\mathbf{l}_0 + \frac{1}{2}(\mathbf{l}_1 - \mathbf{l}_0)\,, \tag{22}$$

the unembeddings are $\mathbf{l}_i + \mathbf{v}$ and the cosine similarity between $\mathbf{l}_0 + \mathbf{v}$ and $\mathbf{l}_1 + \mathbf{v}$ is:

$$\cos(\mathbf{l}_0 + \mathbf{v}, \mathbf{l}_1 + \mathbf{v}) = 1\,. \tag{23}$$

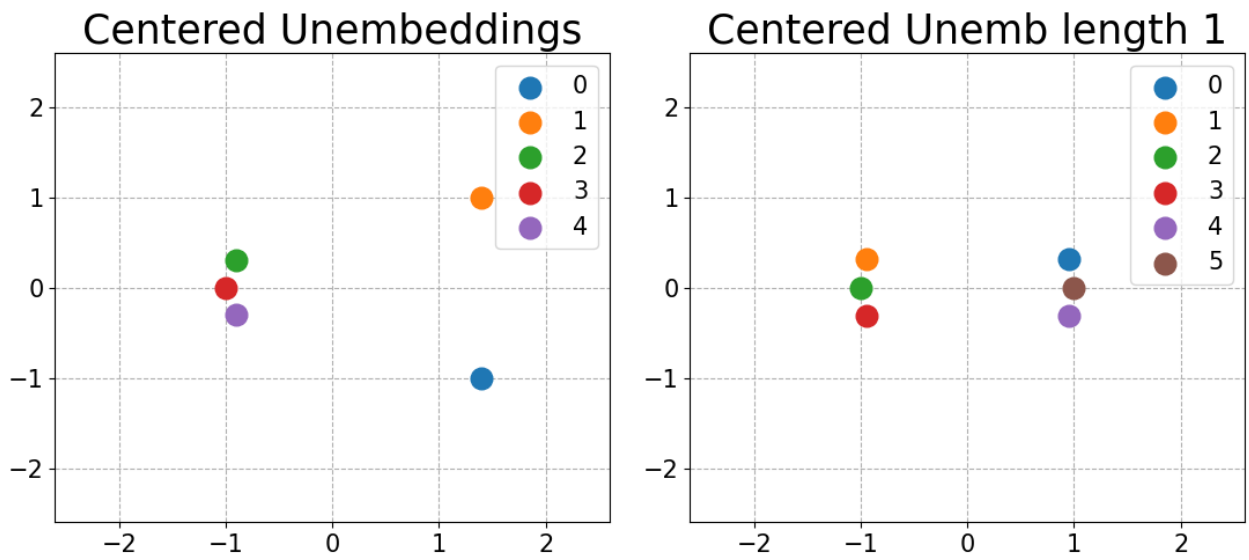

Figure 2. Examples of models with centered unembeddings. In both cases, we see that a high (or low) cosine similarity between unembeddings does not guarantee that the corresponding labels will have high (or low) probability for possible inputs. Left: These unembeddings are centered. The cosine similarity between 2 and 4 is much higher than between 2 and 1, but 2 and 1 can be likely at the same time while 2 and 4 cannot. Right: These unembeddings are centered and have length 1. 1 and 0 can be likely at the same time while 1 and 3 cannot.

C.2. Example using Centered Unembeddings

We here present an example of a model with centered unembeddings illustrated in Figure 2 left. For the five labels the unembeddings are:

$$\mathbf{l}_0 = \begin{pmatrix} 1.4 \\ -1 \end{pmatrix} \ , \ \mathbf{l}_1 = \begin{pmatrix} 1.4 \\ 1 \end{pmatrix} \ , \mathbf{l}_2 = \begin{pmatrix} -0.9 \\ 0.3 \end{pmatrix} \ , \ \mathbf{l}_3 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \ , \ \mathbf{l}_4 = \begin{pmatrix} -0.9 \\ -0.3 \end{pmatrix} \ .$$

These unembeddings are centered since the sum is the zero vector. The cosine similarity between $\mathbf{l}_0$ and $\mathbf{l}_1$ is approximately 0.3 and between $\mathbf{l}_1$ and $\mathbf{l}_2$ it is approximately -0.6, however for both these pairs of labels it is possible to place embeddings which will result in equal probability for both labels. On the other hand, the cosine similarity between $\mathbf{l}_2$ and $\mathbf{l}_4$ is 0.8, but it is not possible for these labels to be tied for the highest probability, since if we place an embeddings between them, the dot-product with $\mathbf{l}_3$ would be higher. This example also shows us that neighbourhoods using cosine similarity do also not correspond to probabilities: The two nearest neighbours of $\mathbf{l}_2$ in terms of cosine similarity are $\mathbf{l}_3$ and $\mathbf{l}_4$, but the the two labels with which $\mathbf{l}_2$ can have tied for highest probabilities are $\mathbf{l}_3$ and $\mathbf{l}_1$.

C.3. Example using Centered Unembeddings of Length 1

We here present an example of a model with centered unembeddings which all have length 1 illustrated in Figure 2 right. For the five labels the unembeddings are:

$$\mathbf{l}_0 = \begin{pmatrix} 0.95 \\ \sqrt{1-0.95^2} \end{pmatrix} \ , \ \mathbf{l}_1 = \begin{pmatrix} -0.95 \\ \sqrt{1-0.95^2} \end{pmatrix} \ , \ \mathbf{l}_2 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \ , \tag{24}$$

$$\mathbf{l}_3 = \begin{pmatrix} -0.95 \\ \sqrt{1-0.95^2} \end{pmatrix} \ , \ \mathbf{l}_4 = \begin{pmatrix} 0.95 \\ \sqrt{1-0.95^2} \end{pmatrix} \ , \ \mathbf{l}_5 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \ . \tag{25}$$

We see that these sum to the zero vector and each of them has norm 1. We see that the cosine similarity between $\mathbf{l}_0$ and $\mathbf{l}_1$ is approximately -0.8, while between $\mathbf{l}_1$ and $\mathbf{l}_3$ it is approximately 0.8. At the same time, $\mathbf{l}_0$ and $\mathbf{l}_1$ can be tied for highest probability, while $\mathbf{l}_3$ and $\mathbf{l}_1$ cannot.

D. The Sigmoid Classifier Case

In multi-label classification, our goal is to classify an input $\mathbf{x}$ by assigning it a subset of non-mutually exclusive labels from a label vocabulary $\mathcal{V}$ of size k. We treat each label y_l, $l \in \mathcal{V}$, as a Bernoulli random variable and we want to model label assignments of the form $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}$ where $\mathcal{Y} = \{+1, -1\}^k$ and a positive label $y_l = +1$ denotes that label y_l was assigned to input $\mathbf{x}$, while a negative label $y_l = -1$ denotes that y_l was not.

Shared Feature Representation Perhaps the simplest model for multi-label classification is to fit an independent binary classifier per label $l \in \mathcal{V}$, parameterised by $\mathbf{g}(l)$, on a shared embedding $\mathbf{f}(\mathbf{x})$. We assign a probability to the presence/absence of each label as:

$$p(y_l \mid \mathbf{x}) = \text{sigmoid}\left(y_l \mathbf{f}(\mathbf{x})^\top \mathbf{g}(l)\right), \quad \text{sigmoid}\,(z) = \frac{1}{1+\exp(-z)}, \tag{26}$$

where we multiply the logits by y_l to express both $p(y_l = +1 \mid \mathbf{x})$ and $p(y_l = -1 \mid \mathbf{x})$ in a single equation. This is possible because $\text{sigmoid}\,(-z) = 1 - \text{sigmoid}\,(z)$.

D.1. Word2Vec as a Sigmoid Classifier

A lot of the language used around embeddings, such as claims that "embeddings with high cosine similarity co-occur often in the data" stem from foundational models of word embeddings, such as count-based co-occurrence models (Deerwester et al., 1990) and early co-occurence based neural network models, such as Word2Vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014). As we highlight in this paper, the co-occurrence rationale does not necessarily hold for softmax embeddings. The multi-label and multi-class setups differ in more than just task formulation: they have different implications for the geometric structure of the unembeddings. Below we connect the skip-gram model of Word2Vec with multi-label classification and sigmoid classifiers, to clarify that the co-occurrence framing is reasonable for sigmoid classifiers.

Word2Vec (Mikolov et al., 2013) popularised neural network word embedding models. The main idea is that words that appear in similar contexts should have similar representations. This is formalised by picking a symmetric token window centred at a given token and learning a model that classifies whether two words co-occur in the token window or not.

Below we clarify that $\text{Skip-gram}\,(\mathbf{W}, \mathbf{Z})$ with a label vocabulary of size k and embedding dimensionality d, is effectively a multi-label classifier parametrised by $\mathbf{W} \in \mathbb{R}^{k \times d}$ together with a learnable set of input embeddings $\mathbf{Z} \in \mathbb{R}^{k \times d}$. More specifically, we interpret the skip-gram model learned via negative sampling (Mikolov et al., 2013, Section 2.2) as multi-label classification, where the positive labels come from the training data and the negative labels are sampled from a proposal distribution. We now show that the negative sampling training objective corresponds to maximising the likelihood of incomplete label assignments. Our goal is to classify which tokens are likely to co-occur together in a token window and which ones are not. We pick a token window size $w = 2r + 1$, $r \in \mathbb{N}^+$, consider the token window $C_t = (c_{t-r}, \ldots, c_t, \ldots, c_{t+r})$ centered around token c_t, and minimise the loss:

$$\mathcal{L}_t = -\sum_{-r \le i \le r,\, i \ne 0} \Big(\log p(y_{c_{t+i}} = +1 \mid c_t) + \sum_{j=1}^{m} \mathbb{E}_{c_j \sim P_k(y)} \big[\log p(y_{c_j} = -1 \mid c_t) \big] \Big), \tag{27}$$

where c_t is the word index for the token at position t in the text, y_{c_i} is the binary label denoting whether token c_i is in the window or not, and m is the number of negative samples sampled from $P_k(c)$, which is a categorical distribution over the label vocabulary.[6] By minimising $\mathcal{L}_t$ in Equation (27) we are maximising the likelihood of an incomplete label assignment: we have $2r$ active labels and $2rm$ inactive labels.[7].

The model is parametrised simply with embedding lookup tables, $\mathbf{W}, \mathbf{Z} \in \mathbb{R}^{k \times d}$ and the labels are scored via a sigmoid activation:

$$p(y_c \mid c') = \text{sigmoid}\left(y_c\, \mathbf{f}(c')^\top \mathbf{g}(c)\right) \tag{28}$$

where $\mathbf{g}(c)^\top = \mathbf{w}_c$ and $\mathbf{f}(c')^\top = \mathbf{z}_{c'}$ are both embedding lookups, *i.e.* rows, of $\mathbf{W}$ and $\mathbf{Z}$.

[6] The paper uses $P_k(y) = U(k)^{\frac{3}{4}}/\mathcal{Z}$ where $U(k)$ is the unigram distribution computed over a corpus.

[7] Strictly, $\mathcal{L}_t$ corresponds to a multi-label objective only when the tokens in C_t combined with the negative samples are distinct.

Decision Regions The decision regions given by $\mathbf{W}$ correspond to full-label assignments. For $\mathcal{L}_t$ to be low, we would expect the decision regions for most contexts seen during training to exist. As such, it makes sense that the decision regions of Word2Vec capture co-occurrence statistics and that this is reflected in the unembeddings.

D.2. Decision Regions of a Sigmoid Multi-label Classifier

We now analyse the decision regions for a multi-label classifier with unembedding matrix $\mathbf{W} \in \mathbb{R}^{k \times d}$. We first consider a single binary classifier. The set of points in embedding space $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^d$ that are assigned label y are the points such that:

$$p(y \mid \mathbf{x}) > p(-y \mid \mathbf{x}) \quad \iff \tag{29}$$

$$p(y \mid \mathbf{x}) > 1 - p(y \mid \mathbf{x}) \quad \iff \tag{30}$$

$$p(y \mid \mathbf{x}) > \frac{1}{2} \quad \iff \quad \text{From Equation (26)} \tag{31}$$

$$\operatorname{sigmoid}\left(y\mathbf{g}(y)^\top \mathbf{f}(\mathbf{x})\right) > \frac{1}{2} \quad \iff \quad \left(\text{sigmoid is a bijection and } \operatorname{sigmoid}(0) = \frac{1}{2}\right) \tag{32}$$

$$y\mathbf{g}(y)^\top \mathbf{f}(\mathbf{x}) > 0 \tag{33}$$

We note that $\operatorname{sigmoid}(0) = \frac{1}{2}$. The decision region for a label combination $(y_1, y_2, \ldots, y_k) \in \{+1, -1\}^n$ is given by $\mathbf{z} \in \mathbb{R}^d$:

$$y_i \mathbf{w}_i^\top \mathbf{z} > 0, \quad \forall i : 1 \leq i \leq k. \tag{34}$$

$p(y_i = j), j \in [-1, 1]$ We note that when $d < k$, there must exist label combinations that cannot be predicted. The label combinations that can be predicted by $\mathbf{W}$ are captured by the sign pattern of $\mathbf{W}$, which we define next.

Definition D.1 (Sign Pattern)**.** We determine the set of feasible label combinations as the sign pattern of $\mathbf{W} \in \mathbb{R}^{k \times d}$ is:

$$\mathcal{S}(\mathbf{W}) = \left\{\operatorname{sign}(\mathbf{W}\mathbf{z}) : \mathbf{z} \in \mathbb{R}^d, \quad \mathbf{w}_i^\top \mathbf{z} \neq 0 \quad 1 \leq i \leq k\right\}, \tag{35}$$

where we disallow zeroes because we want to focus on the decision regions and not parts of the decision boundaries[8], and where the $\operatorname{sign}()$ function is applied element-wise:

$$\operatorname{sign}(z_i) = \begin{cases} + & \text{if } z_i > 0, \\ - & \text{if } z_i < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

We note that our sign pattern definition above extends naturally to matrices that have more columns than rank. If $\mathbf{W}' \in \mathbb{R}^{k \times d'}$, $d' > \operatorname{rank}(\mathbf{W}') = d$, we can truncate the singular value decomposition of $\mathbf{W}' = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ to the top d principal directions without losing any information, and use the resulting matrix instead.

Proposition D.2. $\mathcal{S}(\mathbf{W})$ ***is the set of orthants of*** $\mathbb{R}^k$ ***intersected by the span of the columns of*** $\mathbf{W}$.

The above can be verified by noticing that $\mathbf{W}\mathbf{x}$ is a linear combination ($\mathbf{x}$ are the coefficients) of the columns of $\mathbf{W}$ and the sign function checks which orthant $\mathbf{W}\mathbf{x}$ falls in.

We can also think of these sign vectors in $\mathbb{R}^d$; they are the decision regions created by partitioning feature space with the decision boundaries of the binary classifiers $\{\mathbf{w}_i\}_{i=1}^k$.

The sign pattern is useful as it tells us the set of possible label combinations that can be predicted via argmax prediction for a multi-label classifier parameterised by $\mathbf{W}$. We will now show that the cosine similarity matrix $\mathbf{C}$ determines $\mathcal{S}(\mathbf{W})$. So while we cannot recover the exact probability distributions we would obtain for an input $\mathbf{x}$, **we can recover the decision boundaries of the classifier** by knowing $\mathbf{C}$.

Definition D.3 (Cosine Similarity Matrix)**.** Let $\mathbf{C} = \mathbf{C}(\mathbf{W}) \in \mathbb{R}^{k \times k}$ be the symmetric matrix with entries $C_{i,j} = \cos(\mathbf{w}_i, \mathbf{w}_j)$, $i, j \in \{1, \ldots, k\}$, where:

$$\cos(\mathbf{w}_i, \mathbf{w}_j) = \frac{\mathbf{w}_i^\top \mathbf{w}_j}{\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2}. \tag{37}$$

[8] Allowing logits to be zero would complicate our set notation: our set would also count decision boundaries, when we are only interested in the set of outputs that can be predicted. Moreover, logits that are exactly zero are unlikely to occur, so ignoring the zero case is not an omission that matters in practice.

E. Pairwise Cosine Similarities Determine the Sign Pattern for a Sigmoid Classifier

We recall Lemma 5.3

Lemma 5.3. *Let $\mathbf{W} \in \mathbb{R}^{k\times d}$ with non-zero rows. Then the cosine similarity matrix $\mathbf{C}(\mathbf{W})$ of the unembedding rows determines $\mathcal{S}(\mathbf{W})$.*

Proof. Let us write $\mathbf{C}(\mathbf{W})$ as a product of matrices:

$$\mathbf{C}(\mathbf{W}) = \mathbf{D}\mathbf{W}\mathbf{W}^\top\mathbf{D} = \mathbf{D}\mathbf{G}\mathbf{D}, \tag{38}$$

where

$$\mathbf{D} = \begin{bmatrix} 1/\|\mathbf{w}_1\|_2 & 0 & \cdots & \\ 0 & 1/\|\mathbf{w}_2\|_2 & & \\ \vdots & & \ddots & \\ & & & 1/\|\mathbf{w}_k\|_2 \end{bmatrix}, \tag{39}$$

and $\mathbf{G} \in \mathbb{R}^{k\times k}$ is the Gram matrix of $\mathbf{W}$, i.e. $G_{i,j} = \mathbf{w}_i^\top\mathbf{w}_j$. Our proof now proceeds in three steps by showing incrementally that the product of matrices does not change the sign pattern:

$$\mathcal{S}(\mathbf{W}) \overset{(1)}{=} \mathcal{S}\left(\mathbf{W}\mathbf{W}^\top\right) \overset{(2)}{=} \mathcal{S}\left(\mathbf{W}\mathbf{W}^\top\mathbf{D}\right) \overset{(3)}{=} \mathcal{S}\left(\mathbf{D}\mathbf{W}\mathbf{W}^\top\mathbf{D}\right), \tag{40}$$

where the first equality uses the fact that $\mathbf{W}$ and $\mathbf{W}\mathbf{W}^\top$ have the same column span, the second uses the fact that scaling the columns preserves the column span, and the third uses the fact that scaling the rows by positive scalars does not change the decision regions when we interpret the rows as normal vectors that define the decision boundary of binary classifiers.

1. $\mathcal{S}(\mathbf{W}) = \mathcal{S}(\mathbf{G})$. We will use Proposition D.2, so it suffices to show that span of the columns for the two matrices is equal, *i.e.* $\text{colspan}(\mathbf{W}) = \text{colspan}(\mathbf{G})$. This can be shown in two steps. **Step 1**: each column of $\mathbf{G}$ is a linear combination of the d columns of $\mathbf{W}$, with coefficients given by each column of $\mathbf{W}^\top$. More concretely, the ith column of $\mathbf{G}$ is:

$$\mathbf{G}_{(:,i)} = \mathbf{W}\mathbf{w}_i = \left(\mathbf{w}_1^\top\mathbf{w}_i, \cdots, \mathbf{w}_k^\top\mathbf{w}_i\right)^\top = \sum_{j=1}^{d} w_{ij} \underbrace{\left(w_{1j}, \overset{\top}{\cdots}, w_{kj}\right)}_{j^{th}\text{ column of }\mathbf{W}}. \tag{41}$$

Therefore, $\text{colspan}(\mathbf{W}) \supseteq \text{colspan}(\mathbf{G})$. **Step 2:** To show that $\text{colspan}(\mathbf{W}) = \text{colspan}(\mathbf{G})$, it remains to show that $\dim\text{colspan}(\mathbf{W}) = \dim\text{colspan}(\mathbf{G})$. We first need to show that $\ker\mathbf{W}^\top = \ker\mathbf{W}\mathbf{W}^\top$ so that our result will follow via rank nullity in Equation (44). Forward direction: if $\mathbf{x} \in \ker\mathbf{W}^\top$, then $\mathbf{W}^\top\mathbf{x} = \mathbf{0} \implies \mathbf{W}\mathbf{W}^\top\mathbf{x} = \mathbf{0}$, so $\mathbf{x} \in \ker\mathbf{W}\mathbf{W}^\top$. Reverse direction: suppose $\mathbf{x} \in \ker\mathbf{W}\mathbf{W}^\top$, so $\mathbf{W}\mathbf{W}^\top\mathbf{x} = \mathbf{0}$. Left multiplying by $\mathbf{x}^\top$, we have $\mathbf{x}^\top\mathbf{W}\mathbf{W}^\top\mathbf{x} = 0 \implies (\mathbf{W}^\top\mathbf{x})^\top(\mathbf{W}^\top\mathbf{x}) = 0 \implies ||\mathbf{W}^\top\mathbf{x}||^2 = 0 \implies \mathbf{W}^\top\mathbf{x} = \mathbf{0}$, so $\mathbf{x} \in \ker\mathbf{W}^\top$.

Via rank nullity and since $\mathbf{W} \in \mathbb{R}^{k\times d}$ has k rows, we have:

$$\text{rank}\left(\mathbf{W}^\top\right) + \dim\ker\mathbf{W}^\top = k \tag{42}$$

$$\text{rank}\left(\mathbf{W}\mathbf{W}^\top\right) + \dim\ker\mathbf{W}\mathbf{W}^\top = k. \tag{43}$$

By subtracting Equation (43) from Equation (42) and rearranging terms, we get:

$$\text{rank}\left(\mathbf{W}^\top\right) = \text{rank}\left(\mathbf{W}\mathbf{W}^\top\right) + \dim\ker\mathbf{W}\mathbf{W}^\top - \dim\ker\mathbf{W}^\top \implies (\ker\mathbf{W}\mathbf{W}^\top = \ker\mathbf{W}^\top) \tag{44}$$

$$\text{rank}\left(\mathbf{W}^\top\right) = \text{rank}\left(\mathbf{W}\mathbf{W}^\top\right) \implies (\text{rank}\left(\mathbf{W}^\top\right) = \text{rank}(\mathbf{W})) \tag{45}$$

$$\text{rank}(\mathbf{W}) = \text{rank}\left(\mathbf{W}\mathbf{W}^\top\right) \tag{46}$$

Therefore, $\dim\text{colspan}(\mathbf{W}) = \text{rank}(\mathbf{W}) = \text{rank}\left(\mathbf{W}\mathbf{W}^\top\right) = \dim\text{colspan}\left(\mathbf{W}\mathbf{W}^\top\right) = \dim\text{colspan}(\mathbf{G})$.

From (1) and (2) we have $\text{colspan}(\mathbf{W}) = \text{colspan}(\mathbf{G})$. Because the sign pattern consists of the orthants intersected by the span of the columns (Proposition D.2) and $\text{colspan}(\mathbf{W}) = \text{colspan}(\mathbf{G})$, we have $\mathcal{S}(\mathbf{W}) = \mathcal{S}(\mathbf{G})$.

2. $\mathcal{S}(\mathbf{W}) = \mathcal{S}(\mathbf{GD})$**.** Scaling each column of $\mathbf{G}$ by a scalar does not change $\operatorname{colspan}(\mathbf{G})$.

3. $\mathcal{S}(\mathbf{W}) = \mathcal{S}(\mathbf{DGD})$**.** Multiplying by $\mathbf{D}$ on the left does not change $\mathcal{S}(\mathbf{GD})$, despite the fact that *it does* change the span of the columns. This is because the decision boundary of a binary classifier does not change if we scale the normal vector which corresponds to the classifier by a positive scalar. Sign vectors arise as decision boundaries of multiple such binary classifiers stacked in the rows of $\mathbf{GD}$[9].

Via Equation (38) we therefore have $\mathcal{S}(\mathbf{C}(\mathbf{W})) = \mathcal{S}(\mathbf{DGD}) = \mathcal{S}(\mathbf{W})$. □

F. The Softmax Classifier Case

In multi-class classification, our goal is to classify an input $\mathbf{x}$ from a set of mutually exclusive labels. We let $\mathbf{x} \in \mathcal{X}$ be inputs and $\mathcal{Y} = \{y_1, \ldots, y_k\}$ be a set of k labels. We model the probability for each class as:

$$p(y_i \mid \mathbf{x}) = \operatorname{softmax}(\mathbf{Wx})_i = \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_i))}{\sum_{j=1}^{k} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_j))} \,. \tag{47}$$

F.1. Decision Regions of a Softmax Multi-class Classifier

For the softmax case, the decision region for label y_t is given by embedding vectors $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^d$ for which the probability of y_t is larger than for y_i and satisfy:

$$p(y_t \mid \mathbf{x}) > p(y_i \mid \mathbf{x}) \quad \forall i : 1 \le i \le k, \quad i \ne t \quad \iff \quad \text{(use Equation (47))} \tag{48}$$

$$\frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_t))}{\sum_{j=1}^{k} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_j))} > \frac{\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_i))}{\sum_{j=1}^{k} \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_j))} \quad \iff \quad \text{(positive \& equal denominators)} \tag{49}$$

$$\exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_t)) > \exp(\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_i)) \quad \iff \quad \text{(log is strictly monotonic)} \tag{50}$$

$$\mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_t) > \mathbf{f}(\mathbf{x})^\top \mathbf{g}(y_i) \quad \iff \tag{51}$$

$$\mathbf{f}(\mathbf{x})^\top \big(\mathbf{g}(y_t) - \mathbf{g}(y_i)\big) > 0 \quad \iff \tag{52}$$

$$(\mathbf{w}_t - \mathbf{w}_i)^\top \mathbf{f}(\mathbf{x}) > 0 \quad \forall i : 1 \le i \le k, \quad i \ne t. \tag{53}$$

We note that in contrast to the decision boundaries for the multi-label case from Equation (29), we now take pairwise differences of the rows of $\mathbf{W}$. We will now specify not only the $k-1$ differences between the target class and all other labels, but all pairwise differences, as this will allow us to reuse Lemma 5.3. Let $\mathbf{B} \in \mathbb{R}^{\binom{k}{2} \times k}$ be the matrix whose rows are $\mathbf{e}_i^\top - \mathbf{e}_j^\top$, $1 \le i < j \le k$ in lexicographic order:

$$\mathbf{B} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 1 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & -1 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{pmatrix}. \tag{54}$$

The rows of $\mathbf{B}$ are the normal vectors of the Braid Hyperplane arrangement (Stanley, 2004, Lecture 1), a well-studied object in combinatorics that splits $\mathbb{R}^k$ into $k!$ regions, each corresponding to a ranking of k elements. We note that the set of rankings that can be produced by a low-rank classifier $\mathbf{W}$, is determined by the regions that are reachable in the image of $\mathbf{W}$, which is the span of its columns. We therefore define the set of feasible rankings for a classifier $\mathbf{W}$ as its ranking pattern.

Definition F.1 (Ranking Pattern)**.** We define the ranking pattern of $\mathbf{W}$ as:

$$\mathcal{R}(\mathbf{W}) = \left\{ \operatorname{sign}(\mathbf{BWz}) : \mathbf{z} \in \mathbb{R}^d, \quad (\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{z} \ne 0, \quad 1 \le i < j \le k \right\}, \tag{55}$$

where as in the sign pattern case, we do not consider any points that fall on the decision boundary.

[9] Another way of stating this result is: scaling the normal vectors of a hyperplane arrangement by positive coefficients does not change the arrangement, so the decision regions are preserved.

The ranking pattern is the set of orthants in $\mathbb{R}^{\binom{k}{2}}$ that can be intersected by $\operatorname{colspan}(\mathbf{W}') = \operatorname{colspan}(\mathbf{BW})$. Each sign vector in this set corresponds to a ranking over the k labels via all pairwise comparisons. As such, we have that $\mathcal{R}(\mathbf{I}_k) = \left\{\operatorname{sign}(\mathbf{Bx}) : \mathbf{x} \in \mathbb{R}^k\right\}$, gives us the set of all possible rankings of k elements, where $\mathbf{I}_k$ is the $k \times k$ identity matrix.

Observe that $\mathbf{W}' = \mathbf{BW}$ contains all pairwise differences of the rows of $\mathbf{W}$, where each row of $\mathbf{W}'$ is a binary classifier $\mathbf{w}' = \mathbf{w}_i - \mathbf{w}_j$ which classifies an input as positive if it ranks $y_i > y_j$.

The ranking pattern determines the softmax decision boundaries, as they are a coarsening of the regions that correspond to rankings of the labels: the softmax decision region for class t is the union of all ranking regions that rank class t first Grivas et al. (2022, Section 3.3).

Observe that $\mathcal{R}(\mathbf{W}) = \mathcal{S}(\mathbf{W}')$. We use this to derive the ranking pattern as the sign pattern of $\mathbf{W}'$ using our earlier argument.

Theorem 5.6. $\mathbf{C}(\mathbf{BW})$ *determines* $\mathcal{R}(\mathbf{W})$.

Proof. We apply Lemma 5.3 for $\mathbf{W}' = \mathbf{BW}$, which gives us that $\mathbf{C}(\mathbf{BW})$ determines $\mathcal{S}(\mathbf{BW})$. We then note that $\mathcal{S}(\mathbf{BW}) = \mathcal{R}(\mathbf{W})$.

We spell out more of the details below. Define $\mathbf{G}' = \mathbf{W}'\mathbf{W}'^\top = \mathbf{BWW}^\top\mathbf{B}^\top$. Now, similar to the sigmoid case, scale the matrix appropriately via $\mathbf{D}'$ to obtain the cosine similarity. Define $\mathbf{D}' \in \mathbb{R}^{\binom{k}{2}\times\binom{k}{2}}$ as

$$\mathbf{D}' = \begin{bmatrix} 1/\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2 & 0 & \cdots & \\ 0 & 1/\left\|\mathbf{w}_1 - \mathbf{w}_3\right\|_2 & & \\ \vdots & & \ddots & \\ & & & 1/\left\|\mathbf{w}_{k-1} - \mathbf{w}_k\right\|_2 \end{bmatrix}. \tag{56}$$

Now we have:

$$\mathbf{C}' = \mathbf{D}'\mathbf{W}'\mathbf{W}'^\top\mathbf{D}' = \mathbf{D}'\mathbf{G}'\mathbf{D}' \tag{57}$$

Applying exactly the same arguments as for the sigmoid case above, we know that $\mathbf{C}'$ determines $\mathcal{S}(\mathbf{BW})$, and hence the ranking pattern $\mathcal{R}(\mathbf{W})$. □

G. Two Counter-Examples

G.1. Pairwise Cosine Similarities do not Determine the Ranking Pattern

Theorem 5.5. $\mathbf{C}(\mathbf{W})$ *(and by extension,* $\mathcal{S}(\mathbf{W})$*) does not determine* $\mathcal{R}(\mathbf{W})$.

Proof. We prove this by giving an example of two models with the same cosine similarities between all their unembeddings, but where their possible rankings are different.

Let $\mathbf{W}$ and $\mathbf{W}'$ be the unembedding matrices of two models where the unembeddings are placed at angles $0, \pi/4, \pi/2, 3\pi/4, \pi, 5\pi/4$ and $7\pi/4$. For $\mathbf{W}$ let the lengths of the unembeddings all be 5 (see Figure 3 left), and for $\mathbf{W}'$ let the lengths be 5 for all but the unembedding for label 3, which has length 2 (see Figure 3 right).

The ranking patterns of these two models are different, since model 1 has a region where label 3 is the top 1 label, but the model on the right has no region where label 3 is the most likely. In other words the ranking pattern of model 2 does not include any rankings where label 3 is the top 1, while the ranking pattern of model 1 includes rankings of this type.

This example proves that $\mathbf{C}(\mathbf{W})$ does not determine $\mathcal{R}(\mathbf{W})$, and by extension via Lemma 5.3, $\mathcal{S}(\mathbf{W})$ does not determine $\mathcal{R}(\mathbf{W})$. □

However, given that our counter-example relies on scaling one of the vectors, which works for vector configurations, one may ask whether this still holds for affine configurations. In the next section we show by counter-example that even for the affine case $\mathcal{S}(\mathbf{W})$ does not determine $\mathcal{R}(\mathbf{W})$.

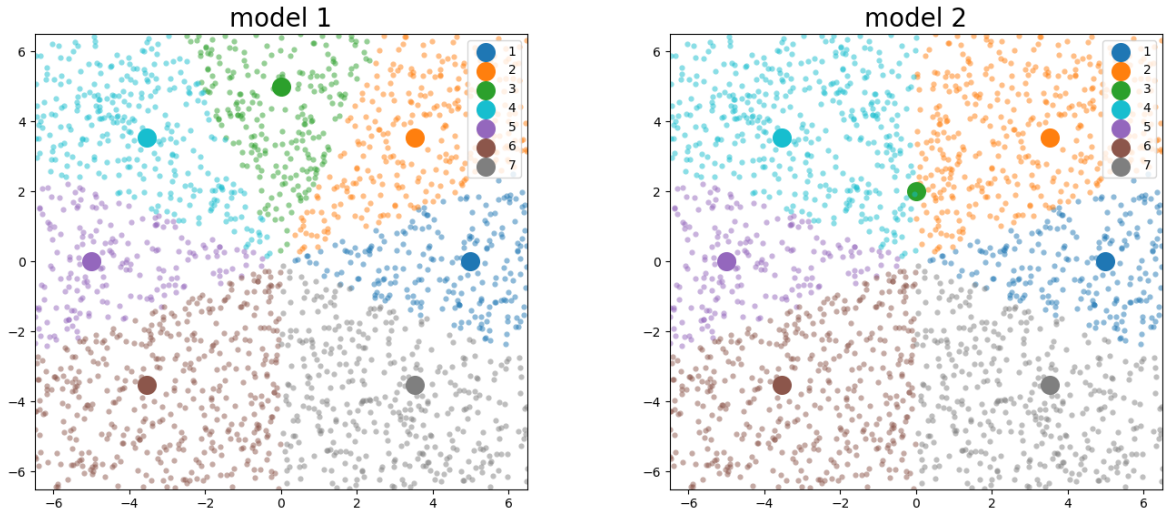

Figure 3. Example of two models with 7 unembeddings placed at the same angles. Big dots are unembeddings. Small dots are embeddings coloured by their top 1 label under the model. The model on the left has a region where label 3 is the top 1 label, but the model on the right has no region where label 3 is the most likely. In other words the ranking pattern of model 2 does not include any rankings where label 3 is the top 1.

G.2. The Sign Pattern does not Determine the Ranking Pattern

Theorem G.1. $\mathcal{S}(\mathbf{W})$ *does not determine* $\mathcal{R}(\mathbf{W})$.

The sign pattern of our vector configuration in the previous example was invariant to scaling each row by a positive scalar, while the ranking pattern was not. We can move from a vector configuration in $\mathbb{R}^d$ to a point configuration in $\mathbb{R}^{d+1}$ by including a constant column. In our example unembedding matrices in Figure 4 below we assume that the first column is constant, as can be seen by the column of ones.

If the sign pattern of the point configuration determined the ranking pattern, then any pair of such unembedding matrices having the same sign pattern would have the same ranking pattern. We show this is not true by providing a counter-example from (Björner et al., 1999, Section 1.10, Figures 1.10.1 & 1.10.2) of two unembedding matrices $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{5\times 3}$, for which $\mathcal{S}(\mathbf{W}_1) = \mathcal{S}(\mathbf{W}_2)$ but $\mathcal{R}(\mathbf{W}_1) \neq \mathcal{R}(\mathbf{W}_2)$.

Sign Pattern We think of the sign pattern of a point configuration $\mathbf{W} \in \mathbb{R}^{k\times(d+1)}$ as follows. When classifying a feature vector $\mathbf{x}$ via $\operatorname{sign}(\mathbf{W}\mathbf{x})$, because the first column is constant, we can rewrite the computation as $\mathbf{W}\mathbf{x} = \mathbf{W}_{(:,1:)}\mathbf{x}_{1:} + \mathbf{1}x_1$, where we are rewriting the matrix-vector multiplication as a linear combination of the columns of $\mathbf{W}$ and using the fact that the first column is a column of ones. Now, from this perspective, we can think of $\mathbf{x}_{1:} \in \mathbb{R}^d$ as a normal vector of an affine hyperplane which is offset by x_1. The sign pattern is then given by thinking of each row of $\mathbf{W}_{(:,1:)}$ as a point in $\mathbb{R}^d$ and checking what side of the affine hyperplane each point can fall on, for all possible affine hyperplanes.

In Figure 4, we show two unembedding matrices, $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{5\times 3}$ that have the same sign pattern but different ranking patterns. To see the sign pattern, we classify the points into positive and negative with any possible affine hyperplane in $\mathbb{R}^2$. For example, if take an affine hyperplane that leaves all points on the same side, we can produce $+++++$, and $-----$ by flipping the hyperplane. Since the point corresponding to the first row is linearly separable from the rest, we

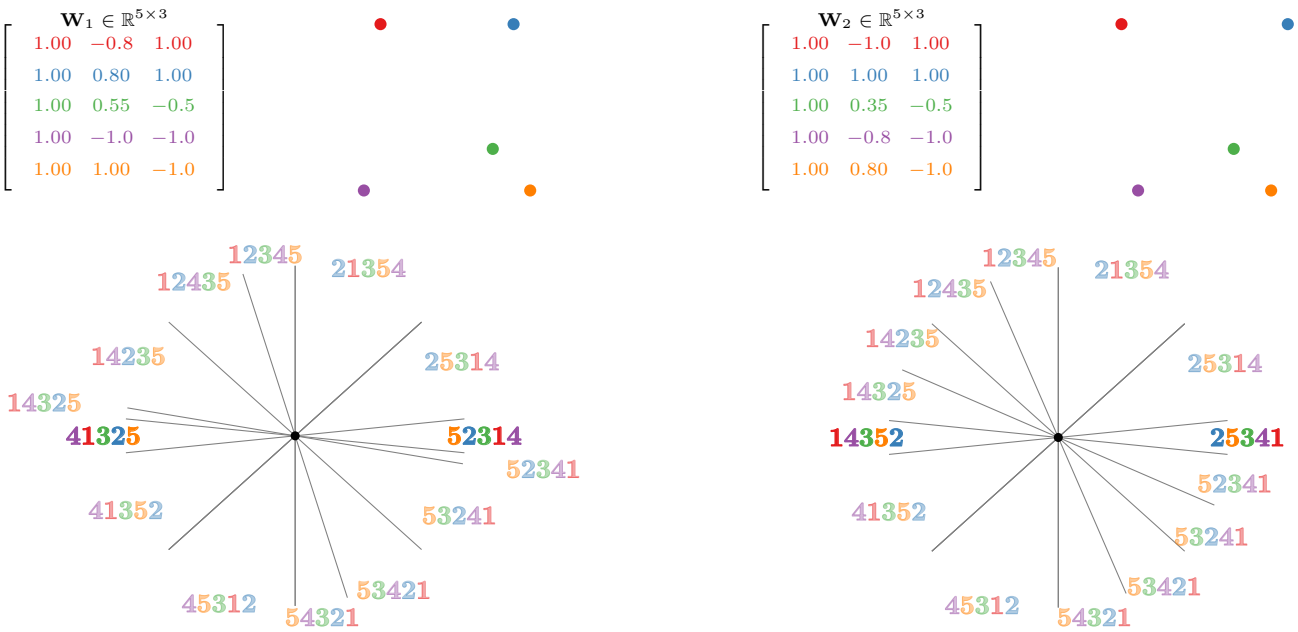

Figure 4. Two point configurations $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{5\times 3}$ that have the same sign pattern but different ranking pattern. We illustrate the two matrices and the corresponding points on the right: each row corresponds to a point of the same colour and has coordinates given by the second and third column. The sign pattern arises by taking all possible affine hyperplanes and separating out the points. Below each point configuration is the partition of the feature space in $\mathbb{R}^2$ into regions that correspond to which ranking would be predicted, where we fade out rankings that appear in both configurations. As can be seen, **4 1 3 2 5** and **5 2 3 1 4** only appear on the left and **1 4 3 5 2** and **2 5 3 4 1** only appear on the right. Therefore, although the sign pattern is the same, the ranking pattern differs, proving that the sign pattern cannot determine the ranking pattern.

can also produce $+----$ and $-++++$. Continuing in this way, the sign pattern for both unembedding matrices is:

$$
\begin{aligned}
\mathcal{S}(\mathbf{W}_1) = \mathcal{S}(\mathbf{W}_2) \supset \{ & +++++, && \text{(All points on one side)} \\
& +----,\ +-+++,\ ++++-,\ +++-+, && \text{(All trapezoid vertices, so } \mathbf{w}_3 \text{ is missing)} \\
& ++---,\ +-++-,\ +++--,\ +--+-, && \text{(All trapezoid edges)} \\
& ++-+- \}, && \text{(Remaining)}
\end{aligned}
$$

where for simplicity we show half the sign vectors, *i.e.* only those that start with a $+$. The remaining negated sign vectors complete the sign pattern.

Ranking Pattern For the ranking pattern, when we compute $\mathbf{BW}$, we note that the constant column cancels to 0 for the pairwise differences of the rows, and as such we can also think of the feasible rankings in $\mathbb{R}^d$ by ignoring the first column. We show the way feature space is partitioned into rankings at the bottom of Figure 4. As can be seen, the ranking pattern for $\mathbf{W}_1$ and $\mathbf{W}_2$ differ: the rankings **4 1 3 2 5** and **5 2 3 1 4** are feasible for $\mathbf{W}_1$ but not for $\mathbf{W}_2$, while the rankings **1 4 3 5 2** and **2 5 3 4 1** are feasible for $\mathbf{W}_2$ but not for $\mathbf{W}_1$.