Unraveling the Complexities of Simplicity Bias: Mitigating and Amplifying Factors

Xuchen Gong*
Machine Learning Department
Carnegie Mellon University
xuchengo@cs.cmu.edu

Tianwen Fu*
Robotics Institute
Carnegie Mellon University
tianwenf@cs.cmu.edu

Abstract

The success of neural networks depends on the generalization ability, while Shah et al. [8] conclude that the inherent bias towards simplistic features, a phenomenon called *Simplicity Bias*, hurts generalization by preferring simple but noisy features to complex yet predictive ones. We aim to understand the scenarios when simplicity bias occurs more severely and the factors that help mitigate its effects. We show that many traditional insights such as increasing training size and increasing informative feature dimensions are not as effective as balancing the modes of our data distribution, distorting the simplistic features, or even searching for a good initialization. Our empirical results reveal intriguing factors of simplicity bias, and we call for future investigations to a more thorough understanding of simplicity bias and its interplay with the related fields.

1 Introduction

The effectiveness of deep neural networks largely hinges on their generalization ability, which is influenced by how closely the model class's inductive bias aligns with the prior assumptions of the real world. Consequently, researchers strive to uncover the underlying model biases and to design algorithms with biases that best match the real-world scenarios.

Conventional wisdom favors simple models with fewer small-magnitude parameters. Approaches following this methodology include parameter sharing [5] and regularization [4, 9, 12]. Moreover, recent work [1, 3, 11] attributes the generalizability of neural networks optimized by stochastic gradient descent (SGD) to an inherent bias towards simpler features, named Simplicity Bias (SB).

However, taking into account the margin of the decision boundary, Shah et al. [8] present the negative impacts of extreme SB: relying exclusively on simple features impairs the robustness of neural networks, leading to poor generalization and out-of-distribution performance. Inspired by the piecewise-linear decision boundary of ReLU-activated multi-layer perceptrons, they proposed synthetic datasets composed of linear slabs and formulated the notion of "simple features" by the number of slabs of the optimal decision boundary that relies on that feature alone. Experiments show that neural networks neglect complex features even if they are more predictive and lead to larger margins, and regularization techniques as well as adversarial training do not mitigate this issue. Methods aiming to resolve the issue, such as gradient regularization for model ensembles [10], lack theoretical guarantee and still have gaps with the optimal performance.

Despite the comprehensive experiments in Shah et al. [8], a number of intriguing or counter-intuitive factors that influence the emergence of simplicity bias remain unrevealed. In this paper, we extend their work by investigating the effect of the feature dimensionality, data distribution, training size, and noisiness of the synthetic dataset to demonstrate the complexity of simplicity bias and the factors

*These authors contributed equally to this work.

Mathematics of Modern Machine Learning Workshop at NeurIPS 2023.

that mitigate or amplify this phenomenon. We hope our work can bring new insights to the discussion of simplicity bias and its relationship to training dynamics and other areas.

2 Related work

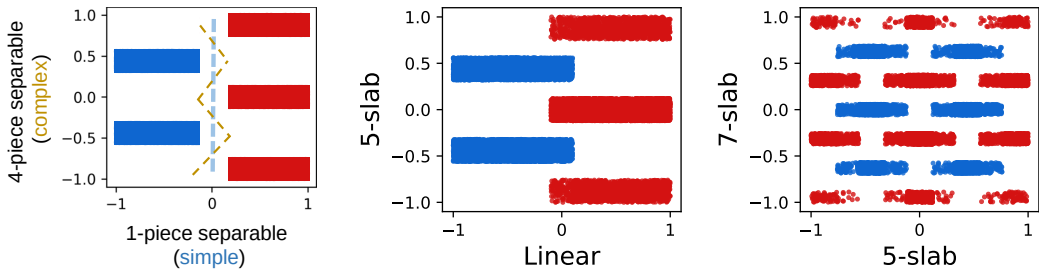
Simplicity bias. Some regard SB as the source of regularization and generalization by showing that the learned high-probability functions of deep neural networks (DNNs) tend to have low Lempel-Ziv complexity [11]. Similar arguments are based on the evidence that a DNN tends to learn simple patterns first [1, 12]. However, opposite viewpoints suggest that the extreme SB leads the neural networks to exclusively rely on the simplest features despite the better predictive power of the complex ones [8]. The resultant small margin classifiers, therefore, exhibit poor generalization and robustness performance.

Resolving simplicity bias. Multiple methods that encourage a deep model to learn complex features are proposed. Assuming that simple solutions are unlikely to be optimal, Dagaev et al. [2] utilizes a low-capacity network, which is only capable of capturing simple patterns, to detect and downweigh the shortcuts learned by a high-capacity network. Another group of methods focuses on balanced learning across features, through either regularization techniques [7] or enforcing disagreement among a group of models to make use of diverse features [6, 10].

3 Preliminaries and setup

Following the seminal prior work on SB [8], we generate multidimensional synthetic data composed of **discriminative while complex** features and **noisy while simple** features as our proxy dataset to understand *when extreme simplicity bias happens and how to mitigate their effects*. On this dataset, we follow their tradition and use four metrics, training accuracy, validation accuracy, S-Randomized accuracy, and S^c-Randomized accuracy to understand a neural network's bias towards simple features.

Data generation. We focus on a binary classification problem and employ the synthetic dataset proposed in [8], where each sample is a d-dimensional vector (i.e., with d features). Among the d features, one feature is designed to be "simple" and the remaining are engineered to be "complex" in the sense that a complex feature requires a more sophisticated discriminatory boundary to separate the data. In our dataset, which is composed of slab groups of data, a boundary's sophistication is measured by its number of linear pieces, as visualized in Fig. 1. The detailed data generation process is specified in Appendix A.



(a) Feature simplicity in `LMS-5` (b) $\hat{\texttt{LMS-5}}$ (c) $\hat{\texttt{MS-(5, 7)}}$

Figure 1: Illustration of feature simplicity and visualization of two dimensions of the generated noisy dataset. In each figure, one dot represents one data sample ($y = +1$ in red, $y = -1$ in blue) and x-axis and y-axis each visualize one feature dimension.

In the whole paper, we follow the tradition and denote `LM-5` as the dataset where one feature dimension forms a 5-slab and the remaining dimensions form 2-slab. `MS-57` represents a dataset where one dimension forms a 7-slab while the remaining form 5-slab. A `\hat` notation denotes a noisy dataset, where the simple feature dimension is noisy and non-discriminative, as shown in Fig. 1b and Fig. 1c. Given this customized noisy dataset, our goal is to probe when a model relies exclusively on one non-informative simple feature rather than the complex ones.

Metrics. Apart from the training and validation accuracy, we also report **S-Randomized accuracy**, the validation classification accuracy when the simple feature values are randomly shuffled, and **S^c-Randomized accuracy**, the validation classification accuracy with the complex feature values randomly shuffled. A low S-Randomized accuracy, therefore, implies reliance on simple features.

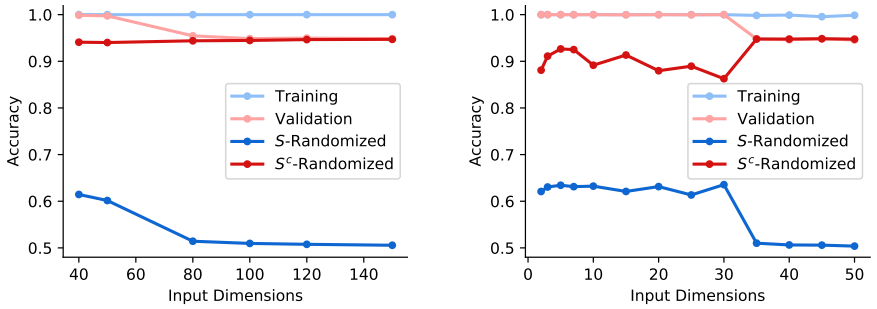
4 Intriguing factors affecting SB

In this section, we conduct controlled experiments by varying one variable of interest at a time, observe its impact on the occurrence of simplicity bias, and discuss the insights. In each experiment, our baseline model and hyperparameters follow the prior work [8] unless otherwise specified. Specifically,

- We experiment on three datasets, $\hat{\text{LMS}}$-5, $\hat{\text{LMS}}$-7, and $\hat{\text{MS}}$-57, each defaultly with 50000 training samples.
- The baseline model is a fully connected neural network with ReLU activations, where the number of hidden layer is 1 and the latent dimension is 300. It is trained with SGD with learning rate $\alpha = 0.3$, weight decay $\lambda = 5.0 \cdot 10^{-4}$, and no momentum.

4.1 More dimensions of predictive features are more of a hindrance

The experiments conducted by Shah et al. [8] are all based on $d = 50$, meaning each sample has one noisy simple feature and 49 predictive complex features. Intuitively, providing more informative features by, for example, increasing d to 150 as in our experiment, should reduce the model's reliance on simple features and thus mitigate simplicity bias. However, we observe that models of the same complexity (*i.e.*, architecture) tend to suffer more from simplicity bias when the data has more informative features, as shown in Fig. 2, although the features are independently generated.



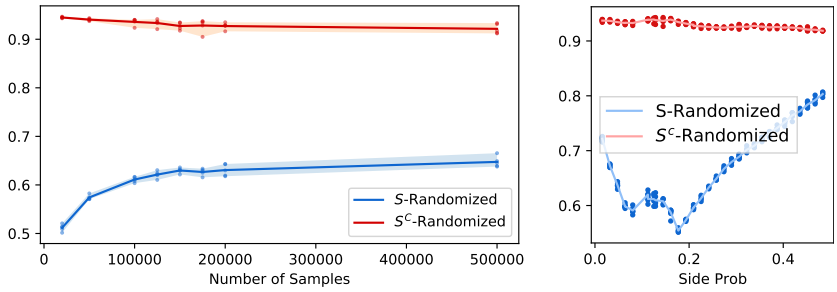
(a) $\hat{\text{LMS}}$-5 (b) $\hat{\text{LMS}}$-7

Figure 2: Results of four accuracy metrics with different input dimensions. As the input dimension increases, the model tends to focus more exclusively on the simple but noisy feature. Dots aligned vertically represent the results of different metrics on the same model.

4.2 Training size matters less than the balanced modes of our data distribution

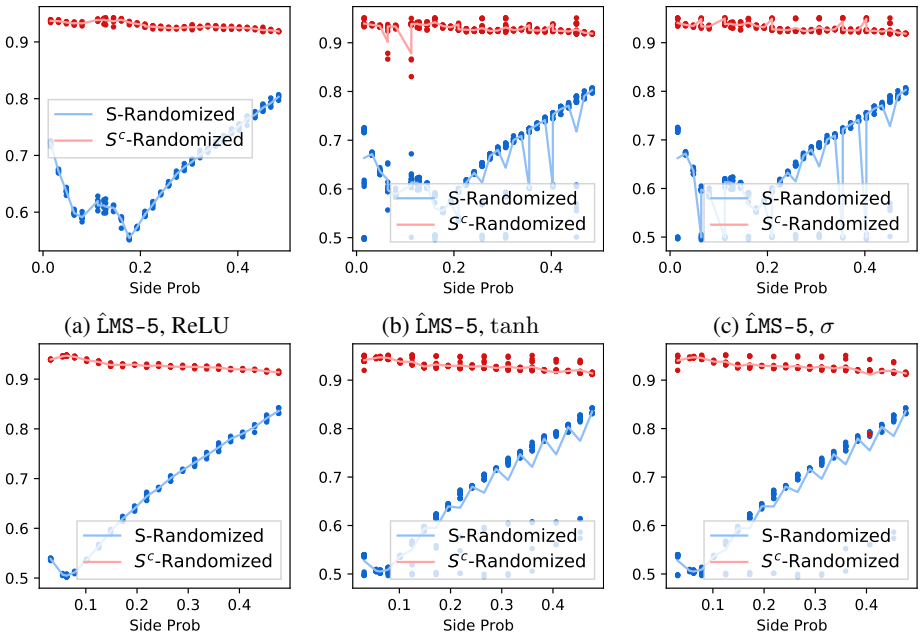
It has been almost a consensus that the size of the training set matters, so increasing training size is expected to mitigate simplicity bias given that the model has a large enough capacity. However, in our experiment (as shown in Fig. 3a), the bias of our models towards simple features stays invariant to the number of training samples, as illustrated by their consistently low S-Randomized accuracy.

Fortunately, though training size does not mitigate SB effectively, balancing the modes of our training data distribution helps. Unlike previous work [8] which distributes a fixed larger probability to sample from the central slabs, we denote the probability of sampling from the slabs on the two farthest sides as $\mathcal{P}^+$ and evaluate the performance under different values of $\mathcal{P}^+$. As shown in Fig. 3b, experiments indicate that reliance on simple features can be largely reduced by increasing $\mathcal{P}^+$, *i.e.*, balancing the distribution modes. Additionally, this phenomenon is also observed when the activation function is changed to $\tanh$ (Fig. 4b and 4e) and sigmoid (Fig. 4c and 4f) apart from the baseline ReLU under the same settings.



(a) Varying number of samples. (b) Varying side proportion.

Figure 3: Accuracies with respect to varying number of samples or changing the side proportion in $\hat{\texttt{LMS}}$-5. We conduct experiments for each setting with different random seeds and plot the average accuracy (lines) and individual results (dots). The figures show that the model's reliance on simple features does not decrease much as the number of samples increases, while varying side proportion plays an impactful role. Each dot represents a model with ReLU activation that is trained until convergence to at least 99.5% training accuracy or reaching the maximum tolerable iterations.



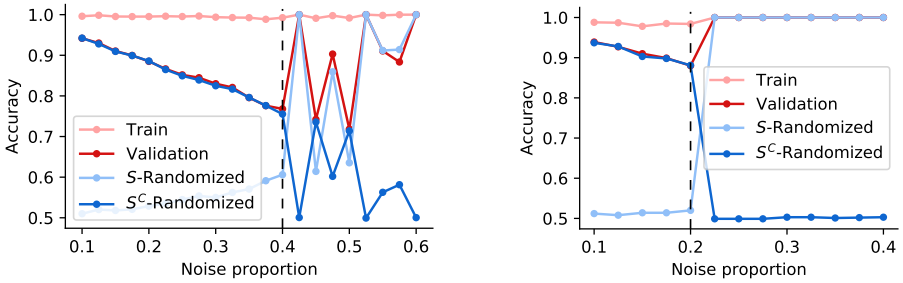
(a) $\hat{\texttt{LMS}}$-5, ReLU (b) $\hat{\texttt{LMS}}$-5, tanh (c) $\hat{\texttt{LMS}}$-5, σ

(d) $\hat{\texttt{LMS}}$-7, ReLU (e) $\hat{\texttt{LMS}}$-7, tanh (f) $\hat{\texttt{LMS}}$-7, σ

Figure 4: Training and validation accuracies with different $\mathcal{P}^+$. $\mathcal{P}^-$ is fixed at the value in Section A.2. We can observe that the undesirable gap in repliance on simple *vs.* complex features can be largely mitigated by sampling data equally from each slab.

4.3 When simple features become noisy, they become less simple

A neural network prefers noisy simple features to predictive complex features, while *how noisy is too noisy*? To investigate how the noisiness of simple features affects a model's preferred features for prediction, we vary the noise proportion (i.e., the proportion of the data points that are noisy) in the range $[0.1, 0.4]$ on $\hat{\texttt{MS}}\texttt{-(5,7)}$.



(a) $\hat{\texttt{MS}}\texttt{-(5,7)}$ with 40k samples. (b) $\hat{\texttt{MS}}\texttt{-(5,7)}$ with 100k samples.

Figure 5: Accuracies *vs.* noise proportion on $\hat{\texttt{MS}}\texttt{-(5,7)}$ of different sizes.

As shown in Fig. 5a, as noise proportion increases, S-randomized accuracy increases, which indicates that the model's predictions rely less and less on the simple features. When simple features become noisy enough, there exists an "**inflection point**," after which the model occasionally resorts to complex features for prediction and achieves a smaller generalization gap. Similar tendencies can also be observed in Fig. 5b, where the model relies entirely on complex features to make predictions when simple features become noisy to some extent.

Therefore, we conclude that a model has SB when data has a small noise proportion; a large noise proportion gives no SB. Moreover, the model's preference for features does not vary smoothly with respect to the noisiness of the simple features but shifts drastically beyond a certain point. Based on this phenomenon, we hypothesize that there is a certain noise level beyond which the models' perception of "feature simplicity" is flipped: The noisier, the more complex the simple feature becomes, which motivates the model to utilize $\texttt{S}^c$ instead.

5 Discussion

Throughout our experiments, we have shown a number of intriguing factors that affect the presence of simplicity bias in neural networks optimized by SGD on the synthetic dataset. More complex but predictive features does not induce the model to utilize them; a balanced dataset can be more useful than a large long-tail dataset; the dramatic change in the feature preferences of models when the simple features are more corrupted may imply the effectiveness of deliberate distortion of simplistic features to boost model performance. Based on these findings, we hope to call for more thorough studies of the factors contributing to simplicity bias, possibly with experiments on realistic datasets.

Besides, the existence of simplicity bias is found to be dependent on the random initialization of models, and the mechanism behind is unclear. Future work may also focus on finding (or disproving the existence of) a region where neural networks are less prone to simplicity bias, or collaborating with related work on training dynamics to mitigate it.

Acknowledgments We thank Jhih-Yi Hsieh for thorough discussion and help with the initial draft.

References

[1] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

[2] Nikolay Dagaev, Brett D Roads, Xiaoliang Luo, Daniel N Barry, Kaustubh R Patil, and Bradley C Love. A too-good-to-be-true prior to reduce shortcut reliance. *arXiv preprint arXiv:2102.06406*, 2021.

[3] Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

[4] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017.

[5] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[6] Matteo Pagliardini, Martin Jaggi, François Fleuret, and Sai Praneeth Karimireddy. Agree to disagree: Diversity through disagreement for better transferability. *arXiv preprint arXiv:2202.04414*, 2022.

[7] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34:1256–1272, 2021.

[8] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[10] Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16761–16772, 2022.

[11] Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.

[12] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

A Synthetic data

A.1 Generation of synthetic data

The synthetic dataset [8] contains N samples $\{(x^{(i)}, y^{(i)})\}_{i=0}^{N-1}$ of d-dimensional features $\{x^{(i)} = (x_0^{(i)}, x_1^{(i)}, \cdots, x_{d-1}^{(i)})\}_{i=0}^{N-1}$ and binary labels $\{y^{(i)}\}_{i=0}^{N-1}$ where $y^{(i)} \in \{-1, +1\}$. Each dimension $X_j \in [-w_j, w_j]$ has width w_j and is a s_j-slab feature, with different s_j's bringing different levels of simplicity. Any consecutive slab pieces are separated by margin γ_j, possibly filled with noise, making up ζ_j proportion of data. For each sample in the dataset, Y is sampled independently from the Bernoulli distribution with $P(Y = -1) = P(Y = +1) = 1/2$. For each data point and its corresponding label $y^{(i)}$, we sample each feature dimension X_j independently based on the feature class.

For each dimension X_j, we first split the interval $[-w_j, w_j]$ into s_j intervals with margin γ_j. Let

$$l_j = \frac{2w_j - 2\gamma_j(s_j - 1)}{s}$$

denote the width of each slab; we may then find the intervals $\{U_j^{(k)}\}_{k=0}^{s_j-1}$ to be

$$U_j^{(k)} = [-w + k(l_j + 2\gamma_j), -w + k(l_j + 2\gamma_j) + l_j]\,.$$

Then, we define the alternating intervals of positive slabs and negative slabs by

$$U_j^+ = U_j^{(0)} \cup U_j^{(2)} \cup \cdots \cup U_j^{2\left(\lceil \frac{s_j}{2} \rceil - 1\right)};\ U_j^- = U_j^{(1)} \cup U_j^{(3)} \cup \cdots \cup U_j^{2\lfloor \frac{s_j}{2} \rfloor - 1}.$$

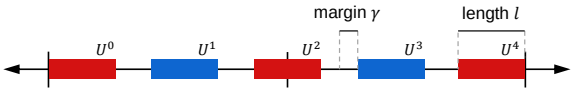


Figure 6: Illustration of a 5-slab feature. The red slabs are in U^+ for positive samples, and the blue slabs are in U^- for negative samples.

Based on the slab intervals, we sample $x_j^{(i)}$ based on the conditional probability density $p(x_j^{(i)}|y^{(i)})$. To ensure equal variances $\mathrm{Var}[X_j|Y = 1] = \mathrm{Var}[X_j|Y = -1]$ for both positive and negative samples, we define the side slabs as

$$U_j^{+,s} = U_j^{(0)} \cup U_j^{2\left(\lceil \frac{s_j}{2} \rceil - 1\right)};\ U_j^{-,s} = U_j^{(1)} \cup U_j^{2\lfloor \frac{s_j}{2} \rfloor - 1},$$

and let $\mathcal{P}_j^+, \mathcal{P}_j^-$ denote hyperparameters denoting the probability of sampling from the slabs on both farthest sides of the positive and negative class, respectively. The probability density functions are shown in Equation 1 and 2.

$$f_{x_j^{(i)}|y^{(i)}=-1}(x) = \begin{cases} \frac{\mathcal{P}_j^-}{2l_j} & \text{if } x \in U_j^{-,s}, \\ \frac{1-\mathcal{P}_j^-}{\left(\lfloor \frac{s_j}{2} \rfloor - 2\right)l_j} & \text{if } x \in U_j^- \setminus U_j^{-,s}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

$$f_{x_j^{(i)}|y^{(i)}=+1}(x) = \begin{cases} \frac{\mathcal{P}_j^+}{2l_j} & \text{if } x \in U_j^{+,s}, \\ \frac{1-\mathcal{P}_j^+}{(\lceil \frac{s_j}{2} \rceil - 2)l_j} & \text{if } x \in U_j^+ \setminus U_j^{+,s}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Noisy features. For the features X_j with noise proportion $\zeta_j > 0$, we generate noise labels Z_j independently of Y by $P(Z_j = 1) = \zeta_j, P(Z_j = 0) = 1 - \zeta_j$. For $Z_j = 0$, X_j is generated according to Y exactly by Equation 1 and 2 above. For $Z_j = 1$, X_j is uniformly distributed in the gaps $[-w, w] \setminus (U_j^+ \cup U_j^-)$ regardless of Y.

A.2 Specification of our datasets

Following the convention in [8], we use $\hat{\texttt{LMS}}$ and $\hat{\texttt{MS}}$ datasets in our experiments:

- $\hat{\texttt{LMS}}$`-k` has each feature containing one 2-slab (*i.e.* linear) block and multiple k-slab blocks. For $k = 5$, $\mathcal{P}_j^+ = 1/4$ and $\mathcal{P}_j^- = 1$ (note that $U_j^{-,s} = U_j^-$ in this case); for $k = 7$, $\mathcal{P}_j^+ = 1/8$ and $\mathcal{P}_j^- = 1/2$. The linear feature is noised with $\zeta_0 = 0.1$, and the more complex k-slab features have 100% predictive power.
- $\hat{\texttt{MS}}$`-(m, n)` has each feature containing one noisy m-slab block and multiple n-slab blocks without noise. $\mathcal{P}$ are the same as in $\hat{\texttt{LMS}}$`-k`.