Exploring the Learning Capabilities of Language Models using LeverWorlds

Eitan Wagner[†] **Amir Feder**[‡] **Omri Abend**[†]
[†]Hebrew University of Jerusalem [‡]Columbia University
eitan.wagner@mail.huji.ac.il

Abstract

Learning a model of a stochastic setting often involves learning both general structure rules and specific properties of the instance. This paper investigates the interplay between learning the general and the specific in various learning methods, with emphasis on sample efficiency. We design a framework called LeverWorlds, which allows the generation of simple physics-inspired worlds that follow a similar generative process with different distributions, and their instances can be expressed in natural language. These worlds allow for controlled experiments to assess the sample complexity of different learning methods. We experiment with classic learning algorithms as well as Transformer language models, both with fine-tuning and In-Context Learning (ICL). Our general finding is that (1) Transformers generally succeed in the task; but (2) they are considerably less sample efficient than classic methods that make stronger assumptions about the structure, such as Maximum Likelihood Estimation and Logistic Regression. This finding is in tension with the recent tendency to use Transformers as general-purpose estimators. We propose an approach that leverages the ICL capabilities of contemporary language models to apply simple algorithms for this type of data. Our experiments show that models currently struggle with the task but show promising potential.[1]

1 Introduction

Many statistical learning settings combine two types of challenges: discovering the underlying persistent *structure* or representation of the problem, and modeling the context-dependent *variability* (Bengio et al., 2013). These two factors differ in their generality – the structure is shared by many cases that might differ in their variability.

For example, assume we want to know how long a typical object will take to reach the ground when dropped from a building in some city. Assuming this knowledge has not been directly reported, we must acquire it from the data. We can conduct experiments by dropping different balls from different buildings. Learning involves acquiring two types of knowledge – one is the physical rules of free fall (e.g., that the mass does not influence the time of the fall), and the other is the distribution of the heights of the buildings in this city (assuming we cannot directly measure the heights). Both types are induced from experiments, but the first is universal, and as such it might already be known based on experiments conducted in a different place.

Recent Large Language Models (LLMs) are used as general purpose learners, as almost any task that does not require multiple modalities can be formulated as text-to-text or text completion. Therefore, the model must learn both the world and stochastic model for effective learning. As typically models are trained with likelihood-based objectives, the models' output confidence for completions should reflect the underlying distribution of the data.

The density estimation task from observations is well-studied in statistics and machine learning. Different methods for parameter estimation differ in their assumptions (or parameterizations). Generally, a model with fewer assumptions can better fit the true unknown) distribution, compared to a model with many assumptions. This is described as having lower *bias*. On the other hand, when a model is more flexible it is more sensitive to noise in the specific training set, hurting generalization. This is described as having higher *variance*. This tradeoff between these two phenomena is known as the *bias-variance tradeoff*. The tradeoff has implications on the *sample complexity* of a model, as high variance might require training sets with more samples.

[1] Code is provided at https://github.com/eitanwagner/leverworlds

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 15458–15468
November 12-16, 2024 ©2024 Association for Computational Linguistics

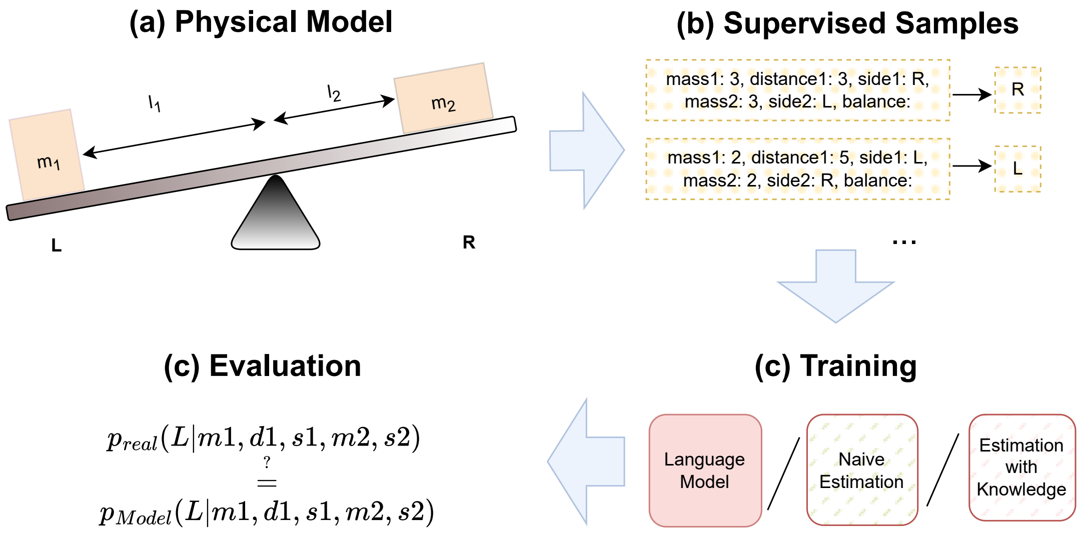

(a) Physical Model
l_1
l_2
m_1
m_2
L
R
(b) Supervised Samples
mass1: 3, distance1: 3, side1: R, mass2: 3, side2: L, balance:
R
mass1: 2, distance1: 5, side1: L, mass2: 2, side2: R, balance:
L
...
(c) Training
Language Model
Naive Estimation
Estimation with Knowledge
(c) Evaluation
$p_{real}(L|m1, d1, s1, m2, s2) \overset{?}{=} p_{Model}(L|m1, d1, s1, m2, s2)$

Figure 1: Overview of our experiments. First, we generate a physical model, then we sample from the model and train a language model to predict the output. We then evaluate the model's probability estimations.

In this work, we investigate the place of LMs within this tradeoff. To show this, we design a simplistic yet rich world of physical models. We train LMs to learn these worlds from samples (see Figure 1 for a schematic overview). We show that even in this simple setting, although better than extremely naïve models, LLMs are substantially less sample efficient than classical statistical estimation methods that have a stronger inductive bias (e.g., logistic regression). Analysis of our experiments shows that this sample inefficiency can be ascribed more to the underlying structure model than to the variability.

We further show that leading off-the-shelf LLMs, like GPT4o[2], fail on this task in an in-context-learning setting. However, inspired by the underlying structure vs. variability distinction, we propose a method for introducing inductive bias into these models by guided construction of a learning pipeline that includes classical models. We find that although challenging for all models, some models show a substantial improvement over others, revealing a promising trend. This further supports the centrality of the distinction.

Our framework, LEVERWORLDS, is also a contribution in its own right. It has many appealing traits: (1) it is based on real physical laws, and is thus not a "toy example"; (2) it includes many variations, all of which are simple to learn when the physical rules are known but complicated otherwise; (3) it follows a generative process, which allows for the generation of arbitrarily large sets of supervised samples, with confidence scores, for training and testing.

Our contributions are: (1) we present a framework for experiments in a physical setting; (2) we show that LMs succeed in learning world models, but they are substantially inferior (in terms of sample complexity) to classical models with stronger assumptions; (3) we show the distinction between learning the world model and learning the latent model; (4) we propose methods for using LLMs in combination with classical models, with promising initial results.

2 Related work

2.1 World Models

In pretrained LLMs. Many works evaluate and explore the extent to which pretrained LLMs encode world models. Abdou et al. (2021) show a correspondence between textual color representations in LMs and a perceptually meaningful color space. Gurnee and Tegmark (2024) show that LLMs learn linear representations of space and time across multiple scales. Vafa et al. (2024) propose evaluation metrics for world model recovery and show that world model representations are still inconsistent even for models with high accuracy in prediction tasks.

[2] https://openai.com/index/hello-gpt-4o/

Learning world models from examples. Some works demonstrate the capabilities of Transformers in learning artificial domains with deterministic rules. Demeter and Downey (2021) and Toshniwal et al. (2022a) show that Transformers trained on chess games can learn to track pieces and predict legal moves with high accuracy. Demeter and Downey (2021) additionally demonstrate the capabilities in baseball game states. Li et al. (2024) show that Transformers trained on Othello games can be linearly mapped to the true board.

Liu et al. (2024b) show that Transformers can successfully learn dynamical systems from in-context sequences alone, thus expanding to probabilistic worlds that describe real events. Patel and Pavlick (2022) show that LLMs have the ability to map descriptions to actions in a grid world, based on in-context examples.

Integrating world models and language. Richens and Everitt (2024) show that robustness under distributional shifts requires an approximate causal model of the data generation. Chen et al. (2023) propose methods to incorporate structures, given as Bayesian Networks, into neural networks and apply these methods to tabular and visual data. Wong et al. (2023) propose a framework that combines neural language models with probabilistic models, enhancing the models' ability to capture and utilize world knowledge effectively. Feng et al. (2024) combine examples and natural language instructions to train a chess model. Carta et al. (2023) propose methods for grounding LLMs in interactive textual environments based on online Reinforcement learning.

2.2 Finetuning and In-Context Learning

A common training paradigm in NLP is to divide training into self-supervised pretraining and task-specific supervised finetuning (Devlin et al., 2019). Brown et al. (2020) showed that large-scale LMs can be used as few show learners, with the task-specific instructions given as a prompt. Performing tasks with prompts only is known as In-Context Learning (ICL) and is gaining popularity (Team et al., 2023; Dong et al., 2024).

ICL is more memory efficient than finetuning and some works argued that it generalizes better (Awadalla et al., 2022). Other works showed that for models with similar sizes, finetuning can generalize well or even better than ICL (Mosbach et al., 2023).

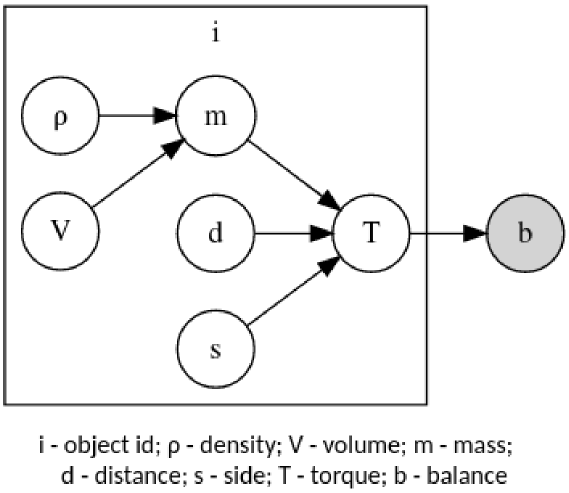

i
ρ
m
V
d
T
b
s
i - object id; ρ - density; V - volume; m - mass;
d - distance; s - side; T - torque; b - balance

Figure 2: Causal graph for balance on a lever. Different worlds differ by the number of objects, by the optional use of density and volume, and by whether the intermediate variables are observed or not.

Despite the power of ICL, Liu et al. (2024a) show that LLMs struggle with long context prompts and degrade significantly when the relevant information is in the middle of a long prompt. Min et al. (2022) analyze the role of demonstrations in ICL. They find that the gold truth labels have little effect and suggest that ICL may not be appropriate when the input-label correspondence is not already captured in the LM.

2.3 Bias-variance Tradeoff

The bias-variance tradeoff is a fundamental concept in machine learning, where a model's capacity to generalize from training data is balanced against its ability to fit the training data accurately. Some recent work, such as Neal et al. (2019), shows that neural networks can defy the traditional bias-variance tradeoff with increased width. Similarly, Dar et al. (2021) discuss how overparameterization in neural networks can lead to better generalization. The tradeoff is important when considering the inductive biases of different models. Parametric models, which assume a specific form for the underlying distribution, often exhibit different bias-variance characteristics compared to nonparametric models, which do not make such assumptions.

3 The LeverWorlds Environment

As empirical validation for our arguments, we experiment with a case in which the world has a simple structure and assess the sample complexity when using language models. Understanding the underlying world model can significantly reduce the effective sample complexity of the task, allowing accurate estimations with a small amount of training data.

We design a framework for generating worlds that enables efficient sampling and estimation. For each generated world, we design models for estimating the distribution of data based on observed samples.

3.1 Setting

For the general framework, we construct worlds by placing weights on a lever. The lever is placed on a fulcrum with some random number of weights on each side. Each world setting is defined by a causal graph (see Figure 2), where different worlds differ by the number of weights (values of i), the distribution of the variables, and which variables are latent. The variables in the model are density (ρ), volume (V), mass (m), distance from the fulcrum (d), and side (s). ρ, V, m and d are real numbers and s can be ± 1. The torques (T) are determined by the other variables according to the formula $T = s \cdot d \cdot m$. For balance (b), $b = 1$ if $\sum_i T_i >= 0$ and otherwise $b = -1$, corresponding the Left and Right, respectively. Masses are determined by the density and volumes, if not latent, according to the formula $m = \rho \cdot V$.

An input x is a sequence of assignments to all the visible variables. We denote the sequence of assignments of the latent variables by l. Given x and l the outcome b is deterministic.

In each given world, sampling is straightforward – we follow the graph and sample the outcome, which is Left or Right. Since the true model is known, we can generate as many samples as we want. This way we can build training sets of arbitrary sizes. During inference, we focus on the output probability given the visible inputs.

We note that each world is defined by two components that must be learned. The first is the general structure of how the outcome depends on a fully observable input, determined by the laws of physics. The second is the case-specific variation which depends on the latent distribution.

The physical model is common to all the settings and is also faithful to the known laws of physics, so models that incorporate general knowledge do not need any samples for this. The latent model is independent between settings and must be learned based on samples in a density estimation process.

3.2 Evaluation

Distribution Similarity The true distribution $p(y|x)$ is known. A learning model yields an estimator $\hat{p}(y|x)$. Evaluation can be done by comparing p and $\hat{p}$.

The main evaluation is simply the distance between the distributions. We decided to use the expected Total-Variation (TV) distance $E_x[d_{TV}(p(\hat{y}|x), p(y|x))] = \frac{1}{2} \cdot E_x[|\hat{p} - p|]$. Other measures, like the Jensen-Shannon distance, gave similar empirical results, so we decided to use TV due to the simplicity in deriving concentration bounds.

Structure Similarity We also include an evaluation that addresses the dependency structure of a learned model by measuring the effect of minimal input changes on the output. For example, given a pair of inputs that have the same values except for the mass of an object on the left, then the outputted probability for "L" must be larger for the input with the larger mass.

Formally, for a set of inputs $\{x\}_{i=1}^{m}$, we generate a modified set $\{x^*\}_{i=1}^{m}$, by randomly choosing one index j in x_i and changing it. Denoting the side of the j-th index by s_j, we define $\Delta x_i := s_j \cdot (x^*_{i_j} - x_{i_j})$ and $\Delta p_i := p(b = L|x^*_i) - p(b = L|x_i)$. We know that, for the ground truth model, $sign(\Delta x_i) = sign(\Delta p_i)$. The structure score for a model M is then defined as

$$Score = \frac{1}{m}|\{i|sign(\Delta p_i^{real}) = sign(\Delta p_i^{M})\}| \quad (1)$$

4 Experiments

In this section, we present the baseline and Transformer models that we use for the experiments. Transformers, as well as some baseline models, are task-agnostic, whereas some baselines leverage task-specific properties.

A different way to describe the difference between models is by the strength of their assumptions about the physical setting (but also about the latent variable, since, e.g., the optimal model assumes it is normally distributed).

4.1 Baseline Methods

Naïve MLE. In this method, we make no assumptions regarding the relationship between inputs. We do, however, assume that we know what the random variables in the input are. Given this, the method independently estimates a Bernoulli distribution for the output, for each possible input, as the frequency of the output in the data.

Formally, the estimator for each input x is

$$\hat{p}(Y=1|x) = \begin{cases} \frac{N_{x,Y=1}}{N_x} & \text{if } N_x > 0 \\ 0.5 & \text{otherwise} \end{cases} \qquad (2)$$

where N_x is the number of samples in which the input is x and $N_{x,Y=1}$ is the number of samples with input x and output 1.

Linear Models. In another baseline, we perform Logistic Regression (LR) for the output given features of the input. We use polynomial features of degree up to 4. The model assumes simple relationships between the output and the inputs and their interactions. Also in this method, we assume that we know what the random variables are. In contrast to the first baseline (which requires estimators for each possible input), the second baseline requires a small number of parameters.

MLE with knowledge of the full structure. In this method, the model knows the underlying rules of the output when all the variables are given. This must include the latent variables which we denote by L. Specifically, the function

$$q(y|x,l) = \mathbb{1}_{\sum_i T_i > 0}$$

is provided, where y is the output, x is an input, and l is some value of the latent variables.

Learning in this model is simply done by estimation of the distribution of the latent variables. Formally, the MLE density estimator for the latent variable l at point c is

$$\hat{p}(l=c) = \frac{1}{N}\sum_i p(l=c|x_i,y_i)$$

where the training data is $\{(x_i,y_i)\}_{i=1}^N$.

The estimator for the output is the marginal $\hat{p}(y|x) = \sum_c \hat{p}(l=c)\cdot q(x,l=c)$.[3]

4.2 Transformer Fine-tuning

Our main investigation addresses the capabilities of general-purpose text models in simple tasks.

Formulation as a text completion task. To formulate the task as language completion, we convert the data into text. We list the visible variables by their names with their values. We use the following template:

> object1 density: <v_1>, object1 volume: <v_2>, object1 distance: <v_3>, object1 side: <v_4>, object1 mass: <v_5>, ... balance: <v_6>

where $v_1, v_2, \ldots$ represent the corresponding values. The values of the side and balance are given as “L” or “R”.

We train generative language models to predict the outcome by generating “L” or “R” and we measure the probability of generating each one.

We can add a prompt to give additional information regarding the setting. However, in our experiments, we found that this prompt has little effect on the performance. We therefore report results without it.

Models. We use the OPT models (Zhang et al., 2022) which come in various sizes. We used the 125m, 350m, 1.3b, and 6.7b parameter models. We used the released weights as initialization and then trained for the task.

For training, we use Low-Rank Adaptation (LORA, Hu et al. (2021)) with rank$=64$. We trained the models for 10 epochs with learning-rate $= 2\cdot 10^{-4}$. We found that although more epochs improve results, the gain is not substantial for larger epoch numbers.

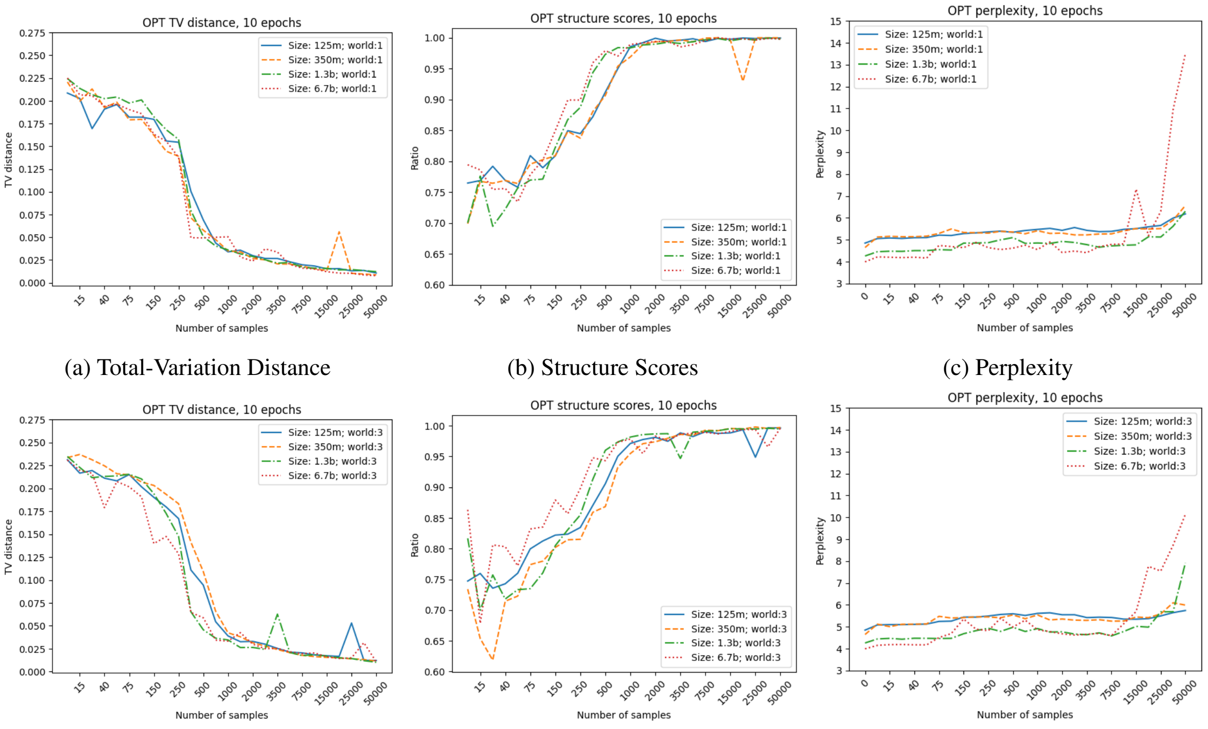
We found that training with randomly initialized model weights is significantly more challenging compared to a pretrained model. We also found LORA to be more stable, compared to finetuning a full model, and it also better preserves the perplexity of language tasks. Therefore we provide results with these settings only.

4.3 Zero-shot Experiments

We evaluated our learning tasks with off-the-shelf models in two zero-shot settings. In one setting we instructed the model to learn the distribution of the data as in-context prompts. In the second setting, we instructed the model to write functions that parse the data and then input the parsed values to a simple logistic regression model.

In-Context Learning. In this setting, the model is given a prompt with a list of observations and is

[3] For simplicity, we assume all variables are discrete. For continuous variables, density should be used instead of mass and the sum should be replaced with an integral.



(a) Total-Variation Distance

(b) Structure Scores

(c) Perplexity

(d) Total-Variation Distance

(e) Structure Scores

(f) Perplexity

Figure 3: Results for OPT models. In the first row are the results for world-1 and in the second are the results for world-3. In cases, we plot the metric as a function of the number of training samples.

asked to give the probability for some test observations.

Assuming a large number of samples is required for sufficient learning, this method is limited to models that are capable of large context windows, such as GPT4. Also, obtaining estimates for all possible inputs (which is required to compute the expected TV distance) is prohibitively expensive. Nevertheless, results for a few sample cases can still provide insight into the learning capabilities of this method.

We ran this setting for two random world settings, each with two random sets of training samples. We put in the context either 10, 100, or 1000 samples (with 1000 for the GPT4 models only). For the exact prompts see Appendix A.1.

Pipeline Learning. Inspired by the distinction between the physical rules and the latent distribution, we propose using LLMs as a step in a workflow pipeline. The model is given some background and examples and is asked to generate a parsing function that will be used with a Logistic Regression model (without polynomial features).

We note that simply parsing the input as a list of values is insufficient for this task since the model does not consider multiplications. On the other hand, calculating the total torque is impossible since some variables are latent.

We tested 3 OpenAI models: gpt-3.5-turbo-0125, gpt-4-turbo-2024-04-09, and gpt-4o-2024-05-13. We sampled 3 worlds, each with 3 different random sample sets, and generated prompts for the models. We also added different levels of hints to assist the model. For more details about the prompts we used see Appendix A.2

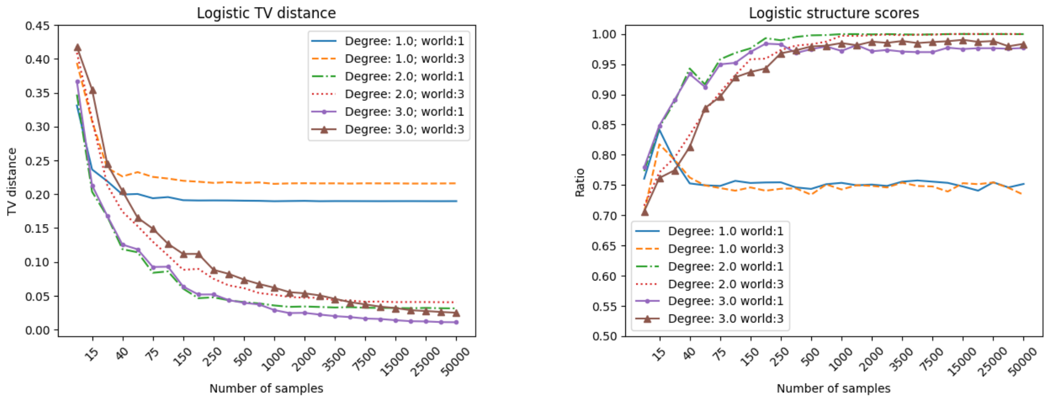
The models generate parse functions which we run on random test sets (using the same sets across all models). We then measure the average TV distance between the predictions and the real probabilities. In cases where the model returns an error, we mark the distance as 1.

5 Results

Here we report and plot the performance of the various models and methods.

We conducted the main experiments on two randomly chosen world settings. The first setting (World-1) has visible variables of *mass1*, *distance1*, *side1*, *mass2*, and *side2*. The second setting (World-3)[4] has visible variables of *density1*, *vol-*

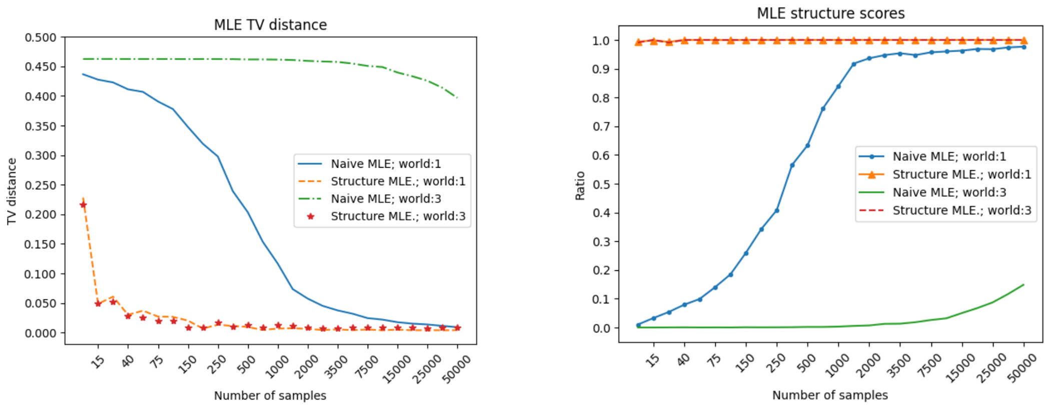
[4] World-1 and World-3 were generated with seed $= 1, 3$, respectively. We conducted the experiments with World-3 for



(a) Total-Variation Distance

(b) Structure Scores

Figure 4: Results for Logistic Regression models.



(a) Total-Variation Distance

(b) Structure Scores

Figure 5: Results for MLE models.

ume1, *mass1*, *distance1*, *side1*, *density2*, *volume2*, *mass2*, and *side2*. Notice that the mass variables are redundant in the second setting. In both settings, there were two objects and *distance2* was latent. In World-1, the mean of the latent variable *distance2* is 2.668; in world-3, it is 3.203. In both cases, the variance is 1.

In Figure 3 we report the TV distances and structure scores for the Transformer models. Additionally, we report the perplexity of the trained Transformers on a portion (first 500 documents) of Wikitext-2.[5] These values serve as indicators of the extent to which the textual pretraining is affected by the task-specific finetuning.

In Figures 4 and 5 we report the TV distances and structure scores for the Logistic Regression and MLE models, respectively.

Model	TV_{pipe}	< 0.1	TV_{ICL}	< 0.1
GPT-3.5	1.	0.	0.424	0.
GPT-4	0.55	**0.22**	**0.207**	0.
GPT-4o	**0.51**	0.037	0.235	0.

Table 1: Results for the zero-shot experiments. TV represents the average TV distance over all samples in all settings and < 0.1 represents the ratio of experiments in which the (average) TV distance was smaller than 0.1. For TV lower is better and for < 0.1 higher is better. Scores are reported for both the In-Context Learning (ICL) and pipeline methods.

Zero-shot scores. Results for the pipeline learning experiment are reported in Table 1.

6 Discussion

As World-3 is clearly more complex than World-1, it is interesting to see how this affects different models. In Naïve-MLE we see a significant performance gap between the settings, with the

variety, as World-2 was similar to World-1.

[5] https://huggingface.co/datasets/Salesforce/wikitext/viewer/wikitext-2-raw-v1

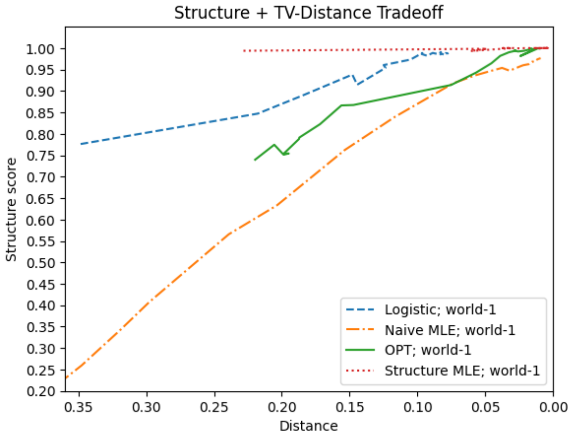


Figure 6: Tradeoff between TV-distance and Structure score. OPT represents the average distances and average scores for all 4 sizes with 5 seeds each.

method only starting to learn with $\approx 10K$ samples in World-3. In Appendix B we derive simple bounds for the expected squared TV distance of this method and show that the extra variables have a substantial effect. In Logistic Regression and OPT models, there is a consistent gap between the two settings but it is not substantial. In Structure-MLE the performance is practically the same. This fits a general trend regarding the provided knowledge about the structure. The more the model is provided with structure, the less the effect of additional variables.

In Figure 6 we plot the tradeoff between the TV distance and the structure score (3.2). Since the Naïve-MLE model makes no explicit assumptions regarding the structure, we can see its curve as representing the structure score that can be achieved without explicit learning. This curve bounds the curves from below. From above, the Structure-MLE bounds all other methods, as it is provided with full structure knowledge by definition. In between, we see that Logistic Regression has a higher structure score (per error) compared to OPT models. The general trend we see is that Transformer models seem to learn the structure to some extent, but it is not as strong as in models that are given stronger assumptions.

In most of our experiments, the models achieve low error. The exceptions are logistic regression with polynomial-degree 1 (for both worlds) and naïve-MLE for world-3. In the first case, the model makes strong assumptions that simply do not fit the world. In the second case, the lack of assumptions regarding the world leads to extremely high sample complexity.

Among different settings of the Transformers, we find that larger models learn with fewer samples. However, smaller models seem to preserve the textual pretraining for longer learning. Additional epochs improve the results, but only up to a certain point.

The trend regarding the provided structure is aligned with the number of parameters in the models. Naïve-MLE has $|states|^{|values|}$ parameters. Structure MLE has 1 parameter. Logistic regression has $|states|^{degree}$. In this respect, the OPT model is an exception, as larger models have better performance.

While our zero-shot experiments generally show low results, they do show promising directions. With In-Context Learning, we find that LLMs, even strong long-context ones, struggle with this task. It seems then that the models implicitly apply simplistic heuristics instead of rigorous analysis. In the pipeline experiment, we see that despite the poor performance, there is a clear hierarchy in which GPT4 and GPT4o clearly outperform GPT3.5. This shows that, to some extent, LLMs can be used as components in a pipeline that uses other models. This type of approach was described in Wong et al. (2023), and we view it as a promising approach for the future.

This observation impacts many highly studied tasks that involve components that do not fall under the description of "natural language", such as chess (Toshniwal et al., 2022b; Feng et al., 2024) and arithmetics (Yuan et al., 2023). Our findings suggest that perhaps tools that were designed for natural language are not optimal for these tasks.

We note that although our experiments address finetuning and inference, the findings are relevant for pretraining too. Our findings show that the training data can contain information, that can be captured by simple models, but LLMs may not capture.

7 Conclusion

In this paper, we presented a novel framework for generating experimental worlds from a common physical setting, with easily manipulable distributions. The framework allows sampling from the ground-truth model and enables carefully controlled experiments in learning the distributions. We applied various methods, from classic learning algorithms to various sizes of Transformers.

The methods range from highly structure-aware to structure-agnostic.

Our findings show that even in a very simple physical setting, models that make stronger assumptions as to the structure present better sample complexity. Specifically, in the given setting, simple structure-based models like Logistic Regression and full structure MLE can be substantially more sample-efficient compared to Transformer models. We further propose an approach to leverage LLMs as part of a pipeline that involves a classic learning algorithm. We show that models still struggle with this task but show a promising trend, as newer models show substantial improvements over older ones.

Limitations

We stress that our experiments with different models differ in the information that is given to the model. For example, the baselines receive tabular data variables whereas the Transformer receives text. Consequently, the comparison is for analysis purposes and is not a strict comparison between methods.

We also note that although inspired by real-world physical settings, the data in our experiments is not distributed in anything like naturally occurring text.

Regarding the zero-shot experiments, we note that the development of our prompt was based on worlds that were not used in the main experiments. However, the worlds are all similar to some extent so the generality of the results is limited.

Acknowledgements

The authors thank Moshe Friedman, Yuli Slavutsky, and Eran Malach for their valuable insights. This research was supported by grants from the Israeli Ministry of Science and Technology, the Israeli Council for Higher Education, and the Israel Science Foundation (grant no. 2424/21).

References

Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. 2021. Can language models encode perceptual structure without grounding? a case study in color. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 109–132, Online. Association for Computational Linguistics.

Richard Arratia and Louis Gordon. 1989. Tutorial on large deviations for the binomial distribution. *Bulletin of mathematical biology*, 51(1):125–131.

Anas Awadalla, Mitchell Wortsman, Gabriel Ilharco, Sewon Min, Ian Magnusson, Hannaneh Hajishirzi, and Ludwig Schmidt. 2022. Exploring the landscape of distributional robustness for question answering models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5971–5987, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3676–3713. PMLR.

Asic Q. Chen, Ruian Shi, Xiang Gao, Ricardo Baptista, and Rahul G. Krishnan. 2023. Structured neural networks for density estimation and causal inference.

Yehuda Dar, Vidya Muthukumar, and Richard G. Baraniuk. 2021. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning.

David Demeter and Doug Downey. 2021. Who's on first?: Probing the learning and representation capabilities of language models on deterministic closed domains. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 210–222, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2024. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36.

Wes Gurnee and Max Tegmark. 2024. Language models represent space and time.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Emergent world representations: Exploring a sequence model trained on a synthetic task.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Toni J. B. Liu, Nicolas Boullé, Raphaël Sarfati, and Christopher J. Earls. 2024b. Llms learn governing principles of dynamical systems, revealing an in-context neural scaling law.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12284–12314, Toronto, Canada. Association for Computational Linguistics.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. 2019. A modern take on the bias-variance tradeoff in neural networks.

Roma Patel and Ellie Pavlick. 2022. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*.

Jonathan Richens and Tom Everitt. 2024. Robust agents learn causal world models.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2022a. Chess as a testbed for language model state tracking.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2022b. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11385–11393.

Keyon Vafa, Justin Y. Chen, Jon Kleinberg, Sendhil Mullainathan, and Ashesh Rambachan. 2024. Evaluating the world model implicit in a generative model.

Lionel Wong, Gabriel Grand, Alexander K. Lew, Noah D. Goodman, Vikash K. Mansinghka, Jacob Andreas, and Joshua B. Tenenbaum. 2023. From word models to world models: Translating from natural language to the probabilistic language of thought.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks?

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

A Prompts For GPT4

A.1 In-Context Learning

For In-Context Learning, we used the prompt:

> Assume we have a model representing a lever on a fulcrum, with two objects on it. The first object is on the right and the second is on the left.
>
> I'll give you a list of partial observations of the states of the model. Notice that some values might be latent. Then I'll ask you to give me the probability for the continuation of some prompt, based on the distribution you can derive from the samples. Be prompt in your answer.
>
> Samples: <list of samples>
>
> Question: I'll give you a list of prompts. Give me a python list with the probabilities of "L", one probability for each

> input. Samples: <list of test samples> Give me a list only with no additional explanations.

A.2 Model Recommendation

Asking the model for a recommended learning method, we used the prompt:

> Assume we have a model representing physical setting. <add the first hint here> Here's a list of partial observations of the states of the model. <add the second hint here> <add the third hint here> Samples: <list of samples>
>
> I want to learn the distribution using Statistics or Machine Learning. Specifically, I want to use Logistic Regression to predict the balance probabilities of new samples. Here is an example of the code:
>
> ```
> ```python
> def fit_lr(X, y):
>     from sklearn.linear_model \
>         import LogisticRegression
>     model = LogisticRegression(
>         max_iter=10000,
>         solver='saga')
>     model.fit(X, y)
>     return model
>
> def predict_lr(model, X):
>     return model.predict_proba(X)
> ```
> ```
>
> Write me python function parse_samples(), that parses each sample and creates a feature function that can be used in the snippet above. Make sure the function is appropriate for both training and inference. Give me code only.

The hints that were (possibly) provided were:

> (1) We have a lever on a fulcrum with objects on the lever.
> (2) Notice that some variables might be latent.
> (3) Notice that the distance of the last object is latent.

B Theoretical Sample Complexities

Here we provide theoretical analysis for the sample complexity of the Naive-MLE baseline. For the loss function, we consider the expected squared TV distance.

The Naive-MLE estimates an independent distribution for y for each set of values of the other variables, x. For each assignment x, assume we have N_x samples, and estimator is $\hat{p}(y = 1|x) = \frac{1}{N_x} \cdot \mathbb{1}_{y=1}$. As a Bernoulli random variable, we know that the variance of $\hat{p}$ is $E[(\hat{p} - p)^2] = \frac{p(1-p)}{N_x}$. This can be bounded by $\frac{1}{4N_x}$.

We have

$$E[TV^2(\hat{p}, p)] = \frac{1}{4} \cdot E[(\hat{p} - p)^2] \leq \frac{1}{16N_x}$$

So, for any $\epsilon^2 > 0$, if we have $N_x \geq \frac{1}{16\epsilon^2}$ samples then the expected squared error will be $\leq \epsilon^2$.

Now, we need to bound the probability of $N_x < \lceil \frac{1}{16\epsilon^2} \rceil \leq N^*$, given a total number of samples N. N_x is distributed as a Binomial random variable with parameters $n = N, p_x = p(X = x)$. Following Arratia and Gordon (1989), we can use the tail bound

$$Pr(N_x \leq N^*) \leq \exp\big(-ND(\frac{N^*}{N}||p_x)\big) \quad (3)$$

for $N^* \leq Np_x$.

Since we assumed i.i.d. for the observed inputs, we can use an identical bound for each case. In the simple case, with 3 input variables with 5 values each (similar to World-1 when combining the distance with the side). For a squared error of less than $\epsilon^2 = 0.05^2$ and get $N^* \geq 25$.

Taking $N^* = 32$, the upper bound in B together with the union bound (for 125 options) gives probability $p > 1 - 0.0092$ for all inputs to have at least 25 samples. This choice for N^* yields $N = 32 \cdot 125 = 4000$ which is similar to the empirical results we got.

This same bound for a case with 8 variables (similar to World-3) this bound goes up to $N = 32 \cdot 5^8 = 1.25 \cdot 10^7$ samples.