



Reward Modeling the World via Text-to-Text Regression

Speaker: Xingyou (Richard) Song (MediaTek, May 15, 2025)



Past + Current Coauthors and Collaborators

Credits

GDM: Vizier/AutoML, Racter Google: ML Efficiency, Platforms, Cloud AI



External Companies: Sakana Al

 \mathcal{F}

 \mathcal{R} R

Academia: CMU, UCLA, Stanford, Cornell Tech









Richard Zhang



Daniel Golovin



Arissa Wongpanich



Daiyi Peng



Sagi Perel

Yutian Chen



Adrian Reyes



Bryan Lewandowski





Mangpo Phothilimthana Jeffrey Yang



Dara Bahri



Yash Akhauri







Eric Tang







2022 - 2023: Year of ChatGPT

- Great for VCs, startups, governments, OpenAI employees
- Al is mainstream
- LLMs are mainstream





2022-2023: Worst year for non-LLM research

- Most **confusing time** for *traditionally objective* fields (e.g. optimization, compilers, hardware, etc.). Questions lingering:
 - How do we remain relevant to LLMs?
 - Should we even care about LLMs?
 - None of the LLM topics are mainstream for these fields



Prompt 1:

"Write a short poem for Valentine's Day."

6 ChatGPT

On this Valentine's Day so bright and so clear, A message of love, whispered soft in your ear. With roses in bloom and chocolates so sweet, Our hearts in a dance, to love's gentle beat.

In every glance, a story untold, In every embrace, a warmth to behold. As stars twinkle above in a sky deep and vast, Our love's a melody that forever will last.

So here's to us, on this day of the heart, A promise, a bond, that we'll never part. With each beat, with each breath, so deeply we'll say, Happy Valentine's, my love, today and always.

Gemini

In a world of fleeting hues, Your love, a vibrant rose, ever true. Amidst the chaos, a calming sight, Your presence, a beacon, burning bright.

Like gentle whispers in the breeze, Your words, a symphony, easing my pleas. A haven of comfort, a tender embrace, In your love, my heart finds its rightful place.

On this day of hearts entwined, My love for you, a love that won't unwind. Through seasons changing, years that unfold, My love for you, a story to be told.



In the garden of love, roses bloom, Their petals soft, their fragrance consumes. Hand in hand, we walk this path, Two souls entwined, our love steadfast.

Your eyes, like stars, guide me through, In your laughter, I find skies so blue. With whispered words and stolen glances, We dance in the moon's tender embraces.

Love, a symphony of heartbeats and sighs, Written in stardust across the night skies. On this day of love, let our souls align, Forever entwined, my Valentine.

A Root Cause: Human Feedback

Is this actually useful?

"Did the world's GDP actually change?"

- Sam Altman (Lex Fridman Podcast)

Original ChatGPT tasks:

- Rewrite email
- Tell jokes
- Random chats
- Write poetry

write a poem about pizza

1

Oh, pizza, circle of delight, A canvas painted, day and night. From simple cheese to toppings grand, A culinary, warm command.

The dough, a base, both soft and thin, Or thick and chewy, where to begin? The sauce, a crimson, sweet embrace, A tangy, herby, warm space.

The mozzarella, melted, white, A creamy, stretchy, pure delight. Pepperoni's spicy, curled embrace, Mushrooms earthy, find their place.

RL Fine-Tuning

Step 1

Collect demonstration data and train a supervised policy.

Step 2

Collect comparison data and train a reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.



RL-HF: RL from *Human Feedback*

Humans dictate subjective ratings:

- Creativity + personality
- Safety
- (Human-verifiable) Factuality

Great for Human Interaction:

- Writing prose (poetry, essays)
- Conversations
- Likeability / Human Values

$y\in\{0,1\} ext{ or } [0,1]$

Poor For:

- Experiment Prediction
- Business / Objective Metrics
- Forecasting

Opinion: Has Human Feedback Been Overdone?





Human Feedback Criticisms Over Time



Andrej Karpathy 🤣 @karpathy · Aug 8, 2024 # RLHF is just barely RL ø ..

Reinforcement Learning from Human Feedback (**RLHF**) is the third (and last) major stage of training an LLM, after pretraining and supervised finetuning (SFT). My rant on **RLHF** is that it is just barely RL, in a way that I think is not too widely Show more

Reward Modeling: Which board seems better for white?



"AlphaGo-Human": Optimize only on human ratings of board??



Superhuman LLMs via True RL

"ChatGPT is a local minimum" - David Silver



Welcome to the Era of Experience

"quantities such as cost, error rates, hunger, productivity, health metrics, climate metrics, profit, sales, exam results, success, visits, yields, stocks, likes, income, pleasure/pain, economic indicators, accuracy, power, distance, speed, efficiency, or energy consumption."

Richard Sutton @ @RichardSSutton · Apr 12 The short paper "Welcome to the Era of Experience" is literally just released, like this week. Ultimately it will become a chapter in the book 'Designing an Intelligence' edited by George Konidaris and published by MIT Press. goo.gle/3EiRKIH

Welcome to the Era of Experience

David Silver, Richard S. Sutton*

Abstract

We stand on the threshold of a new era in artificial intelligence that promises to achieve an unprecedented level of ability. A new generation of agents will acquire superhuman capabilities by learning predominantly from experience. This note explores the key characteristics that will define this upcoming era.

O 197

11 20K

口土

 $\bigcirc 4$

tl 51

RL from Ground Truth Rewards: Superhuman Reasoners

BENCHMARK	GEMINI 1.5 PRO 002	GEMINI 2.0 FLASH EXP	GEMINI 2.0 FLASH THINKING EXP 01-21
AIME2024 (Math)	19.3%	35.5%	73.3%
GPQA Diamond (Science)	57.6%	58.6%	74.2%
MMMU (Multimodal reasoning)	64.9%	70.7%	75.4%
Evaluated with the Gemini API using greedy sampling.			









RL on CUDA Kernel Latency

Example of RL applied on real world cost functions



Our research interns present: Kevin-32B = K(ernel D)evin

It's the first open model trained using RL for writing CUDA kernels. We implemented multi-turn RL using GRPO (based on QwQ-32B) on the KernelBench dataset.

It outperforms top reasoning models (o3 & o4-mini)!



RL from World Feedback?

Superhuman ML Scientists: Train on experiment outcome feedback

Superhuman Chemical Designer: Train on making chemicals

Superhuman Chip Designer: Train on making expensive chip designs

Rewards are expensive!!



Interesting(?) Prediction From <u>AI Safety Forum</u>

During most of 2024, these RL training runs cost around \$1 million, sometimes \$10 million. These runs were little more than exploratory. But by 2025, the researchers at OpenEye (and across the world) knew they had found the secret sauce. It was time to scale up.

Over the first half of 2025, \$10 million RL training runs turn into \$50 million runs, and then to \$100 million runs. While U2 could do a bit of data munging and run small experiments, this new model – the model researchers are calling U3 – is changing the daily lives of the technical staff.

U3 is like a blazing-fast intern, and engineers are learning how to wrangle its sleepless energy. Researchers flick through terminals, giving terse commands, like a CEO orchestrating staff over Slack channels.

By October 2025, U3 is writing almost all of the code at OpenEye. Researchers are almost never bottlenecked by implementation. More than ever, compute is the lifeblood of AI development, and the 'bottleneck' is deciding how to use it.

If instructed to, U3 can run experiments, but U3 doesn't have taste as refined as human researchers at OpenEye. It struggles to prioritize between research ideas, so humans still decide where to bore into the vast fields of algorithms to mine efficiency improvements.

But these researchers are working long hours to put themselves out of a job. They need AI agents that can think ahead, so engineers train agents to forecast. They hold out training data before 2024, instructing models to ponder for hours to predict events in 2025. Then, they apply the same trick as before, distilling pondering into a gut reaction. Forecasting ability is a broad foundation. The researchers build specialized ML research skills on top of it, training U3 to predict the results of every ML paper and ML experiment ever recorded.

Reward Models from World Feedback Lens

Reward Models are more broadly, LM regressors over objectives:

$$f:\mathcal{X}
ightarrow\mathbb{R}$$

Current focus on **human feedback**, but should be extended to **any world feedback**.

Examples:

- Outcomes of expensive experiments
- Metrics over production systems
- Environmental feedback



Different Names for Different Communities

LLM Lingo

- Reward Model
- AutoRater
- Verifier

Objective Function Lingo

- Regression
- Performance Prediction
- Surrogate



Regression with Language Models

Description

Regression in experimental design.

x = Tabular Hyperparameters

- Float (ex: Learning Rate)
- Category (ex: SGD or Adam)
- Integer (ex: Number of Layers)

y = Scalar Metric (ex: Accuracy)

$$y = f(x)$$

Prompt (X):

I'm training a **ResNet-52 on CIFAR-10** with hyperparameters **batch size=256** and learning_rate=0.01 with SGD, over 100 epochs. Predict accuracy?

Target (Y):

0.9 0.8 Accuracy 9.0 2.0 ResNet-50 ResNet-101 ResNet-152 0.4 DenseNet-121 0.3 DenseNet-169 DenseNet-201 0.2 Ô 10 20 50 30 40 Epoch

Benefits

LLM Community

- Reward Models Simulating Systems / Nature
- More precise reward modeling







World News

Objective Metric Community

- Flexible text-based regression
- Multi-task + meta-learning
- Easy + Scalable



Tabular Data



Architecture



(x_train, y_train),(x_test, y_test) = mnist.load_data() x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
 tf.keras.layers.Flatten(),
 tf.keras.layers.comse(512, activation=tf.nn.relu),
 tf.keras.layers.comse(512, activation=tf.nn.softmax)
 tf.keras.layers.comse(10, activation=tf.nn.softmax)

/ odel.compile(optimizer='adam', loss='sparse_categorical_crossentropy metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5 model.evaluate(x_test, y_test)



Regression with Chip Design?



X can be:

- Layout / Floorplan
- Hyperparameters / Configuration files
- Associated code

Y can be:

- Power
- Efficiency
- Cost
- Size
- Reliability





Ex: Test-Time Compute

Verification improves inference correctness, what if we had a "world verifier"?



Ex: Could we speedup program search?



Original Prototype: OmniPred (T5)

Google DeepMind

OmniPred: Language Models as Universal Regressors

Xingyou Song*1, Oscar Li**², Chansoo Lee
1, Bangding (Jeffrey) Yang³, Daiyi Peng¹, Sagi Perel
1 and Yutian Chen 1

¹Google DeepMind, ²Carnegie Mellon University, ³Google

*Equal Contribution. †Work performed as a student researcher at Google DeepMind.

Regression is a powerful tool to accurately predict the outcome metric of a system given a set of parameters, but has traditionally been restricted to methods which are only applicable to a specific task. In this paper, we propose OmniPred, a framework for training language models as universal end-to-end regressors over (x, y) data from arbitrary formats. Using data sourced from Google Vizier, one of the largest proprietary blackbox optimization databases in the world, our extensive experiments demonstrate that language models are capable of very precise numerical regression using only textual representations of mathematical parameters and values, and if given the opportunity to train at scale over multiple tasks, can significantly outperform traditional regression models.

1. Introduction

5

202

Jan

30

[cs.LG

arXiv:2402.14547v6

Regression is a fundamental task for experimental design, in many domains such as hyperparameter tuning, computer software, industrial engineering, and chemical discovery. The goal of regression is to predict a metric *y* of a general system given a set of input features *x*. Such regressors can later be used for various applications, such as offline optimization (Kumar et al., 2022; Trabucco et al., 2022), online optimization (Cai et al., 2020), low-cost benchmarking (Eggensperger et al., 2015; Zela et al., 2022) and simulation (Hashemi et al., 2018; Kaufman et al., 2021; Mendis et al., 2019).



Figure 1 | Overview of our method. Using heterogenous offline (x, y) evaluation data collected from a variety

High-Level Goal: Regression via Text!

Simple, general, scalable regression method

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued

If you are working on incremental ideas, be aware that their usefulness depends on their complexity. A method that slightly improves on the baseline better be very simple, otherwise no one will bother using it—not even you. If it gives a 10% improvement, it better be 2 lines of code, whereas if it's a 50% improvement, it can add 10 lines of code, etc. (I'm just giving these numbers for illustration, the actual numbers will obviously depend on the domain.) **– John Schulman**

Main Punchline

If you have (x,y) pairs from **ANY** regression tasks:

- 1. Convert x and y to text
- 2. Next token prediction (prompt = x, target = y)
- 3. Profit! Simple + Scalable



X can be **ANYTHING**

YrSold	SaleType	SaleCondition	MasVnrArea_na	GarageYrBit_na	LotFrontage_na	OverallQual	OverallCond	MasVnrArea	BsmtFin SF1
2009	WD	Normal	False	False	False	-0.8188	-0.5244	-0.2393	-0.0105
2007	WD	Normal	False	False	False	0.6476	-0.5244	-0.5666	-1.0042
2006	WD	Normal	False	False	False	0.6476	-0.5244	0.0715	-1.0042
2008	WD	Normal	False	False	True	0.6476	3.0697	0.5787	0.9901
2009	WD	Normal	False	False	False	-0.8188	0.3741	-0.5666	- <mark>1.004</mark> 2
2007	WD	Normal	False	False	False	-0.8188	0.3741	-0.5666	-1.0042
2008	WD	Normal	False	False	False	-0.0856	-2.3215	2.7276	-0.1807
2006	WD	Normal	False	False	True	-2.2851	-1.4229	1.2877	-1.0042
2009	WD	Normal	False	False	False	-1.5519	-1.4229	-0.5666	0.1666

Tabular Data



Computation Graph

import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
tf.keras.layers.Flatten(),
tf.keras.layers.bense(512, activation=tf.nn.relu),
tf.keras.layers.Doropout(0.2),
tf.keras.layers.bense(10, activation=tf.nn.softmax)

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)

Code

// Information about a Job that is connecting to the message WorkerJobInfo { // The borg user. string user = 1;

// The name of the borg job. This will be the sam // rumbo.aggregated_borglet_measures.* tables. It // as StrCat(service_name, ".", job_name), where : // found in the borg file that launched the job. string job_name = 2;

// The cell this job is running in.
string cell = 3;

// A description of the client that is being used
// Examples:
// C++, Python, HPTuner, etc.

string client_name = 4;

// Field describing specialized tuning tasks, e.g. string application_type = 5;

// This holds a collection of metadata items. It c; // metadata (when Smetadata.trial_id is unset) or T; // Smetadata.trial_id is set), message BulkMetadataUpdatat { message KeyValuePlus { // Which Trial should this metadata be attached // When Strial_id is unset, this carries Studyco optional ini32 trial id = 1:

// The metadata itself.
KeyValue key_value = 2;

3

Protobuf

Serialization for Tabular Regression (Ex: AutoML)

Standard "Prefix LM" Training: (Prompt, Response)

- X: Function input
- M: Metadata describing function
- Y: Objective value

	Language Model Textual Representation	
x	<pre>batch_size:128,kernel:'rbf',learning_rate:0.5,model:'svm',optimizer:'sgd'</pre>	
	title:'classification',user:'some-person',description:'spam detection',	
m	objective:'accuracy'	
y	<+><1><2><3> <e-2></e-2>	

Serialization (X)

- Natural Language / JSON
- Raw value, no normalization!

Language Model Textual Representation					
x	<pre>batch_size:128,kernel:'rbf',learning_rate:0.5,model:'svm',optimizer:'sgd'</pre>				
	title: classification, user. some person, description: spam detection,				
m	objective: 'accuracy'				
y	<+><1><2><3> <e-2></e-2>				

Serialization (M)

- Natural Language
 - Shove in anything
- Conditions distribution
 - Important: Username, title, objective



Serialization (Y)

- Fixed-length custom tokens
 - (sign, mantissa, exponent)
- Restricted decoding (logit masking)
 - Always output correct tokens
- Raw value, no normalization!

0	Language Model Textual Representation		
x	<pre>batch_size:128,kernel:'rbf',learning_rate:0.5,model:'svm',optimizer:'sgd'</pre>		
\overline{m}	<pre>title:'classification',user:'some-person',description:'spam detection',</pre>		
y	<+><1><2><3> <e-2></e-2>		

Regression Paradigm Change

Dynamic Input Spaces: (X) serialization ignores search space bounds

Multitask: Just look at (M)

No Tensorization: Avoid fixed-length embeddings

No Rescaling/Normalization: Avoid numerical instability issues

Regressor	Dynamic Input Spaces?	Can Multitask?	Tensorize?	Rescale?
MLP	No	Only fixed spaces	Yes	Yes
Tree-based	No	Only fixed spaces	Yes	Optional
Gaussian Process (GP)	No	Only fixed spaces	Yes	Yes
GNN / Transformer / RNN	No	Only fixed domains	Yes	Yes
OMNIPRED (Ours)	Yes	Yes	No	NO

Basic Language Model

Basic 12-layer T5X Encoder-Decoder (200M Params)

- No English pretraining.
- ~8 GPU for Training, 1 GPU inference

≡ 🜍 google-research / t5x	Q Type / to searc	ch	<u>></u> <u>+ -</u> O n
<> Code • Issues 79 1 Pull re	equests 94 🖓 Discussions	🕑 Actions 🕕 Secu	rity 🗠 Insights
C t5x Public	🖈 Edit Pins 👻	⊙ Watch 38 ▾ 💡	? Fork 280 🔹 🏠 Star 2.4k
₽ main • ₽ ©	Q Go to file t	+ <> Code •	About
T5X Team and t5-copybara Sw	i 🚥 🗙 7b24d66 · last week	① 1,051 Commits	No description, website, or topics provided.
.github/workflows	Fixes the github clu installation is	5 months ago	🛱 Readme
docs	Change get_vocabularies referen	2 months ago	柸 Apache-2.0 license ∽ Activity
t 5x	Switch the deprecated jax.tree	last week	E Custom properties
	Internal change	3 years ago	☆ 2.4k stars

Example Dataset: Google Vizier (Hyperpameter Optimization)

Using Google Vizier Data

Convenient source of regression data from:

- AutoML (Hyperparameter tuning)
- Chemistry / Biological
- Production (Ads Bidding)

Property	Statistic	
# Studies	<i>O</i> (70M+)	
# Trials	<i>O</i> (120B+)	
# Distinct Users	<i>O</i> (14K)	



Vizier Metadata

Metadata: Title, Owner, Description, Objective Name, Free-form text

Transferability sources:

- Single user, similar experiments
- Different user, similar experiments
 - Ex: ResNets on CIFAR10
- Similar params, different experiments
 - Ex: "learning_rate"
- Description / Free-Form Metadata
 - Ex: Associated code / file locations


Domain	X	Μ
Google AutoML	<pre>batch_size: 128 model_type: REDACTED activation_fn: "tanh" batch_norm: "True" dropout: 0.143 embedding_combiner: "mean" gradient_clip_norm: 1.63e+03 num_hidden_layers: 1 hidden_units[0]: 359 optimizer_type: "AdamOptimizer" beta1: 0.9 beta2: 0.999 learning_rate: 0.926 nlp_vocabulary_strategy: "adjusted_mutual_info" vocabulary_strategy: "adjusted_mutual_info"</pre>	<pre>title: "n-w597ng99i7lj0-q40zcboilea7l" user: REDACTED description: "" objective: "val_categorical_cross_entropy' amc_model_version: REDACTED task_type: "multi_class_classification"</pre>
	"adjusted_mutual_info"	

Domain	X	Μ
Init2Winit	dropout_rate: 0.6	title: "d_sp1-lm1b_trfmr-b1024-2021aug20"
	decay_factor: 0.0379	user: REDACTED
	label_smoothing: 0.378	description: ""
	lr_hparams.base_lr: 0.00285	objective: "valid/ce_loss"
	lr_hparams.decay_steps_factor: 0.854	
	lr_hparams.power: 1.94	
	opt_hparams.one_minus_momentum: 0.0557	

Domain	X	Μ
Protein Design	p00000000:"9" p00000001:"16" p00000002:"1" p00000003:"11" p00000004:"16" p00000006:"9" p00000006:"0" p00000007:"14" p00000047:"13"	<pre>title: "871cac30956711eab5bef371aa1bb25a" user: REDACTED description:"" objective:""</pre>

Domain	X	M
Vertex AI Text	universal_model_type:	title: "20220228-621c9aea-0000-2c94"
	"single_dense_feature"	user: REDACTED
	token_model_type: "cnn"	description: REDACTED
	token_bow_combiner: "sqrtn"	objective:"micro-auc-pr-label0_label"
	token_model_type: "bow"	<pre>atc_study_dataset_tag:""</pre>
	rand:0	atc_study_notes:""
	batch_size: 4	
	convnet: "2:3:4*100pa"	
	dropout_keep_prob: 1	
	hidden_layer_dims: 50	
	max_length: 1.54e+03	
	<pre>max_num_classes_for_per_class_metric: 0</pre>	
	<pre>max_token_vocab_size: 1e+05</pre>	
	<pre>merge_word_embeddings_vocab_size: 1e+05</pre>	
	token_freq_cutoff: 1	
	tokenizer_spec: "delimiter"	
	word_embedding: REDACTED	
	word_embedding_dim: 100	

Experiments

Key Questions:

- Can LM simultaneously regress on multiple-tasks?
 - Over different spaces and numeric scales?
- How is multi-task training useful?
 - \circ Why would we train on f'(x) if we're eval-ing on f(x) only?
- Can fine-tuning deal with unseen tasks?
 - Does pretrained knowledge carry over?



Simultaneous Regression (Synthetic Functions)

Is LM capable of high-precision, simultaneous regression?



Simultaneous Regression (Real World)

What about real-world objectives?



Multi-task Training (AutoML Data)

If I only eval on f(x)...

Does training on f'(x) help?

Data Scaling Laws Occur!



(| Lower is better)

Evaluation Across Domains

Single-task (apples-to-apples):

- Beats GP, Random Forest
- Beats MLP sometimes!

Multi-task beats Single-task

- Beats baselines most times
- Win by data scaling

Name	# Studies	Avg. TpS	Avg. SS
BBOB	1 M	30	4.0
Bid Simulation	22K	698	4.6
Google AutoML (Internal)	540K	250	(3.3, 29.9)
Init2winit	2K	176	3.6
Protein Design	54K	584	125.6
Vertex AI AutoML (Tabular)	1.4M	88	(4.6, 42.4)
Vertex AI AutoML (Text)	544K	118	56.0

Multi-task LM vs Single-task Baselines



(| Lower is better)

Local Finetuning Experiments

<u>Classic GPT-Trick:</u> Pretrain then Finetune

$$\theta_{ft} = \theta_{pt} - \nabla \ell_{(x,y)} \sim \mathcal{D}_{train}(\theta)$$

(| Lower is better)



Eval on unseen Vizier studies

- Studies after March 31, 2023
- From distinct users

Local Finetuning Experiments

Positive Transfer

$$\theta_{ft} = \theta_{pt} - \nabla \ell_{(x,y)} \sim \mathcal{D}_{train}(\theta)$$

 Pretrained knowledge carries over to new studies



Negative Transfer...?

• Sometimes pretrained knowledge is worse...?

Example + Future Applications

XLA Latency Prediction Problem

Predict Latency of XLA compilation graph

• Baseline: GNN + hardcore feature engineering





XLA Latency Prediction: No Feature Engineering Required



<+><8><6><4><2><E1>

Grand Goal? Predict Software + Hardware Objectives

Absorb all experiment data so far

Predict Latency / Efficiency, etc. given:

- Code, Compiler logs, Hyperparameters
- Hardware Layout (e.g. Chip Design)





Grand Goals: Predicting AI Scientist / AutoML Outcomes

Prompt = Entire Codebase

Outcome = Experiment measurement





Text-to-text regression: Powerful, Simple, Scalable

Input can be anything:

- Configuration Files
- Graphs
- Combinatorics
- Language
- Code
- Images



Generalize Targets

Y not just floats, also language outcomes?

• Predicting Scientific Experiments

More broadly, text-based world modeling?

• Ex: Genie 2 (Video to Video tokens)



Large language models surpass human experts in predicting neuroscience results

Xiaoliang Luo Akilles Rechardt, Guangzhi Sun, Kevin K. Nejad, Felipe Yáñez, Bati Yilmaz, Kangjoo Lee, Alexandra O. Cohen, Valentina Borghesani, Anton Pashkov, Daniele Marinazzo, Jonathan Nicholas, Alessandro Salatiello, Ilia Sucholutsky, Pasquale Minervini, Sepehr Razavi, Roberta Rocca, Elkhan Yusifov, Tereza Okalova, Nianlong Gu, Martin Ferianc, Mikail Khona, Kaustubh R. Patil, Pui-Shee Lee, ... Bradley C. Love + Show authors

Nature Human Behaviour (2024) Cite this article

"AGI" of Regression



Thank you!

Unused Slides (Appendix)

Ablation Papers

Google DeepMind

Understanding LLM Embeddings for Regression

Eric Tang^{*1}, Bangding Yang² and Xingyou Song³ ¹Stanford University, ²Google, ³Google DeepMind

*Work performed during the Google DeepMind Academy Program.

With the rise of large language models (LLMs) for flexibly processing information as strings, a natural application is regression, specifically by preprocessing string representations into LLM embeddings as downstream features for metric prediction. In this paper, we provide one of the first comprehensive investigations into embedding-based regression and demonstrate that LLM embeddings as features can be better for high-dimensional regression tasks than using traditional feature engineering. This regression performance can be explained in part due to LLM embeddings over numeric data inherently preserving Lipschitz continuity over the feature space. Furthermore, we quantify the contribution of different model effects, most notably model size and language understanding, which we find surprisingly do not always improve regression performance.

1. Introduction and Related Work

Regression is a fundamental statistical tool used to model the relationship between a metric and a selected set of features, playing a crucial role in various fields, enabling predictions, forecasting, and the understanding of underlying relationships within data. Traditional regression techniques often rely on handcrafted features or domain-specific knowledge to represent input data. However, the advent of Large Language Models (LLMs) and their ability to instead process semantic representations of text has raised the question of whether regression can instead be performed over free-form text.

Previous works have predominantly examined the topic of LLM-based regression through *decoding*, i.e. generating floating point predictions using token-based sampling. For example, (Song et al., 2024) examines the case when the model is fully accessible and fine-tunable against data, while (Vacareanu et al., 2024) study the ability of *service-based* closed-source LLMs such as GPT-4 using in-context learning.

One understudied case however is the use of service-based LLM embeddings - fixed vector representations derived from pre-trained (but frozen) language models, which are ubiquitously offered among most LLM services (Anthropic, 2024; Google, 2024; OpenAI, 2023). Although they are used frequently in recent applications such as retrieval (Karpukhin et al.



 $\label{eq:Figure 1} Figure \ 1 \ | \ Rugged \ surface \ of \ a \ 5D \ Sphere \ function \\ when inputs \ are \ represented \ as \ Gemini \ embeddings \ of \\ dimension \ 6K+, \ post-processed \ by \ t-SNE \ into \ 2D \ space. \\$

Decoding-based Regression

Xingyou Song^{*1} and Dara Bahri^{*1} ¹Google DeepMind *Equal Contribution.

Language models have recently been shown capable of performing regression tasks wherein numeric predictions are represented as decoded strings. In this work, we provide theoretical grounds for this capability and furthermore investigate the utility of causal auto-regressive sequence models when they are applied to any feature representation. We find that, despite being trained in the usual way - for next-token prediction via cross-entropy loss - decoding-based regression is as performant as traditional approaches for tabular regression tasks, while being flexible enough to capture arbitrary distributions, such as in the task of density estimation.

1. Introduction

Despite being originally developed for the purpose of text generation and chat applications, large language models (LLMs) have recently been applied for new applications, one of which is regression tasks, and more broadly the prediction of numeric outcomes. Vacareanu et al. (2024) have shown service-based LLMs such as ChatGPT and Gemini capable of in-context regression with performance comparable to that of traditional regression methods such as random forests, while Song et al. (2024) have shown smaller models can be trained specifically on large amounts of regression data to even outperform baselines.

For input-output pair (x, y), where x is a feature vector and y is a real number, a regression model's performance is determined by two fac-



Figure 1 | Given any feature representation $\phi(x)$, we attach a decoding-based head to output predictive distribution $p_{\theta}(y|x)$.

Input-Side: LLM embeddings are Lipschitz continuous Embedding("0.1") is close to Embedding("0.11")



Output Side: Decoding Numbers

Is it data efficient + accurate?

- Cross-entropy, not MSE
- Decoder, not Pointwise head

Analyze Decoder head attached to any

feature representation



Older Riemannian Head

Softmax over lots of bins

Inefficient: Learn 1000 bins

Decoder uses (10 Vocab)^(3 Tokens)



$$Z_{\theta}(x,a) = z_i$$
 w.p. $p_i(x,a) := \frac{e^{\theta_i(x,a)}}{\sum_j e^{\theta_j(x,a)}}.$

General Tokenization

Normalized (Tree-based) in [0,1]:

0.<b1><b2><b3><b4><b5><b6>...

Unnormalized (Sign-Exponent-Mantissa):



Theory

Binary tokenization

K = Number of bits, Bins = 2^{K}

N = Training data size

Define risk R as the mean integrated squared error between true density f and its estimator $\hat{f}_N(Y_1, \ldots, Y_N)$:

$$R(f,\widehat{f}_N) = \mathop{\mathbb{E}}_{Y_1,\ldots,Y_N \sim f} \int_0^1 \left(f(y) - \widehat{f}_N(y) \right)^2 dy.$$





$$R\left(f,\widehat{f}_{N}^{k}\right)\approx\frac{2^{-2k}}{12}\int_{0}^{1}f'(y)^{2}dy+\frac{2^{k}}{N},\quad\forall k\leq K.$$



Decoder generalizes better than Riemann

K is high, N is low: Riemann struggles to to learn more bins

Decoder generalizes better!

Riemann fits to Signal + Noise

Decoder fits to Signal



Unnormalized Tokenization vs Pointwise Head



Real World (OpenML) Regression Tasks





Data Efficiency: Real World Regression

Decoder more data-efficient than Riemann

Sometimes outperforms even Pointwise!



Density Estimation: Toy Tasks

Temperature Sampling:

Most unbiased



Better Modeling for Real-World Data

MDN (Gaussian Mixtures) Unstable

Decoders: More reliable

Dataset	MDN	UD	ND	R
Airfoil	$0.12{\pm}0.1$	$0.40{\pm}0.0$	$0.34{\pm}0.0$	$1.33{\pm}0.1$
Bike	4.59 ± 0.9	$0.12{\pm}0.0$	$0.10{\pm}0.0$	$0.36{\pm}0.0$
Elevators	$0.30{\pm}0.4$	$0.15{\pm}0.0$	0.13±0.0	$1.12{\pm}0.0$
Gas	$0.68{\pm}0.2$	$0.02{\pm}0.0$	0.02 ± 0.0	$0.20{\pm}0.1$
Housing	$0.22{\pm}0.1$	$0.41{\pm}0.0$	$0.38{\pm}0.0$	$1.56{\pm}0.2$
Kin 40K	$7.49{\pm}0.8$	$0.19{\pm}0.0$	$0.12{\pm}0.0$	$0.39{\pm}0.0$
Pol	$1.49{\pm}0.4$	$0.01{\pm}0.0$	$0.01{\pm}0.0$	$0.18{\pm}0.0$
Protein	$1.07{\pm}0.4$	0.34±0.0	$0.41 {\pm} 0.0$	$1.55{\pm}0.0$
Pumadyn32nm	$0.69{\pm}1.2$	$0.55{\pm}0.0$	$0.58{\pm}0.0$	$2.32{\pm}0.0$
Wine	$0.05{\pm}0.1$	$0.24{\pm}0.0$	$0.21{\pm}0.0$	$1.67{\pm}0.1$
Yacht	$0.21{\pm}0.1$	$0.39{\pm}0.0$	$0.23{\pm}0.0$	$1.29{\pm}0.3$

Table 2: Lower (\downarrow) is better. Avg. NLL (\pm StdDev) of test examples on UCI datasets over 10 train-test splits. Abbreviations: (UD, ND) = (unnormalized, normalized) decoders respectively; R = Riemann.

Gradients vs In-Context

ICL not only way to absorb (x,y).

Gradients:

- Unbounded limit for absorbing (x,y) pairs
- Easy JSON formatting, no (X) compression

Analogous to MLP vs GP


Best y-tokenization?

- (Default) Separate Sign and Digit-by-Digit: <+><1><2><3><4><E-2>
- Merged Mantissa: <+1234><E-2>
- Exponent Before Mantissa: <+><E-2><1><2><3><4>

	AutoML		BBOB	
Tokenization Method	Single-Task	Multi-Task	Single-Task	Multi-Task
Default	0.21	0.15	0.17	0.01
Merged Mantissa	0.73	0.15	0.41	0.01
Exponent Before Mantissa	0.24	0.15	0.17	0.01

Table 6: Mean Study Error (\downarrow) comparisons between different tokenization methods.

Importance of sampling aggregation?

	Mean Study Error (\downarrow)		
Empirical Aggregation Method	AutoML (Full 540K)	BBOB (Full 1M)	
Median (default)	0.15	0.01	
Max-likelihood	0.22	0.01	
Mean	0.23	0.01	



When does Multi-task training help?

Intuitively when eval task has low training data.

