TRAIN ON VALIDATION (TOV): FAST DATA SELECTION WITH APPLICATIONS TO FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

State-of-the-art machine learning often follows a two-stage process: (i) pre-training on large, general-purpose datasets; (ii) fine-tuning on task-specific data. In fine-tuning, selecting training examples that closely reflect the target distribution is crucial. However, it is often the case that only a few samples are available from the target distribution. Existing data selection methods treat these target samples as a validation set and estimate the effect of adding or removing a single sample from the training pool by performing inference on the validation set.

We propose a simpler and faster alternative that inverts the usual role of train and validation: we perform inference on the training pool before and after fine-tuning on the validation set. We then select samples whose predictions change the most. Our key insight is that the training samples most affected by fine-tuning on a small validation set tend to be the most beneficial for reducing test loss on the target distribution. Experiments on instruction tuning and named entity recognition tasks show that, in most cases, our method achieves lower test log-loss than state-of-the-art approaches. We support our findings with theoretical analysis.

1 INTRODUCTION

While large language models (LLMs) are pretrained on internet-scale datasets, their downstream performance can be heavily dependent on the instruction-tuning stage in which they are fine-tuned on instruction/output pairs (Ouyang et al., 2022; Zhou et al., 2024; Longpre et al., 2023; Chung et al., 2024). These datasets are significantly smaller and are often gathered by using multiple heterogeneous sources. Instruction tuning becomes even more difficult when targeting a specialized use case (Wang et al., 2023). More generally, scarcity of domain-specific data is a ubiquitous challenge when fine-tuning foundation models.

This paper presents an easy-to-implement and low-complexity method for selecting a training dataset of prescribed size from heterogeneous sources to maximize the test time performance on the target distribution. Our method is motivated by the theory of influence functions (van der Vaart, 2000) yet avoids the computational burden of computing influence functions. We validate this approach on two token-based learning tasks, instruction tuning and named entity recognition (NER), and show that in most cases it outperforms state-of-the-art data selection baselines. To illustrate its broad applicability, we show that it yields interesting results even for a simple logistic regression example (see Appendix D).

To formalize the problem, assume access to two datasets: a small dataset from the target distribution $\mathbb{P}$ on $\mathcal{Z}$ and a larger one from possibly heterogeneous data sources. We refer to the dataset from the target as the 'validation set' $\boldsymbol{Z}^{\text{val}} := (\boldsymbol{z}_1^{\text{val}}, \dots, \boldsymbol{z}_{m_{\text{val}}}^{\text{val}})$ where $\boldsymbol{z}_i^{\text{val}}$ are i.i.d. samples from the target distribution $\mathbb{P}$, and to the larger heterogeneous dataset as 'training pool' $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$, where $\boldsymbol{x}_i \in \mathcal{Z}$. In general the distribution of the training pool differs from $\mathbb{P}$. Our goal is to minimize the test error on the target distribution with respect to the model parameters $\boldsymbol{\theta} \in \mathbb{R}^p$:

$$R(\boldsymbol{\theta}) := \mathbb{E}[\ell(\boldsymbol{\theta}, \boldsymbol{z})] , \tag{1}$$

where $\ell : \mathbb{R}^p \times \mathcal{Z} \to \mathbb{R}$ is a loss function. A separate target-distribution test set $\boldsymbol{Z}^{\text{tst}}$ (separate from $\boldsymbol{Z}^{\text{val}}$) is used to estimate $R(\boldsymbol{\theta})$ after fine-tuning.

We aim to achieve this by training (or fine-tuning) the model on a subset $S \subseteq [N]$ of the training pool, e.g. running stochastic gradient descent (SGD) with respect to the empirical risk:

$$\widehat{R}_S(\boldsymbol{\theta}) := \frac{1}{|S|} \sum_{i \in S} \ell(\boldsymbol{\theta}, \boldsymbol{x}_i)\,. \tag{2}$$

Let $\hat{\boldsymbol{\theta}}_S$ be the outcome of running SGD (or any specific training algorithm) on $\widehat{R}_S(\boldsymbol{\theta})$. We want to select the subset S (given a constraint on its size $|S|$) so that $\hat{\boldsymbol{\theta}}_S$ achieves a small test loss on the target distribution, i.e. as to minimize $R(\hat{\boldsymbol{\theta}}_S)$.

1.1 Train on validation: motivation and algorithm

To select the most helpful examples at model $\boldsymbol{\theta}$, we might score training examples by the decrease in validation loss induced by a single gradient step with respect to that example, then select those with the highest scores. Computing these scores directly requires $N{+}1$ full evaluations over the validation set. We derive an efficient approximation to these scores.

Consider a single gradient step with respect to a training example $\boldsymbol{x}$:

$$\boldsymbol{\theta}_{\boldsymbol{x}} = \boldsymbol{\theta} - \eta \nabla \ell(\boldsymbol{\theta}, \boldsymbol{x})\,. \tag{3}$$

The corresponding change in loss for a validation example $\boldsymbol{z}$, $\ell(\boldsymbol{\theta}, \boldsymbol{z}) - \ell(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{z})$, can be approximated by a first-order Taylor expansion:

$$\ell(\boldsymbol{\theta}, \boldsymbol{z}) - \ell(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{z}) \approx -\langle \nabla \ell(\boldsymbol{\theta}, \boldsymbol{z}), \boldsymbol{\theta}_{\boldsymbol{x}} - \boldsymbol{\theta} \rangle = \eta \langle \nabla \ell(\boldsymbol{\theta}, \boldsymbol{z}), \nabla \ell(\boldsymbol{\theta}, \boldsymbol{x}) \rangle, \tag{4}$$

where the last step follows from Eq. (3). Pruthi et al. (2020) approximate the scores by computing gradients for each training and validation example and taking their dot products; Xia et al. (2024) extend this to token-based learning. Our method diverges from these approaches: it requires no per-example gradients.

Note the right-hand side is symmetric in $\boldsymbol{x}$ and $\boldsymbol{z}$. In other words, the decrease in loss on $\boldsymbol{z}$ from a step on $\boldsymbol{x}$ is mirrored by the decrease in loss on $\boldsymbol{x}$ from a step on $\boldsymbol{z}$. Our method exploits this train–validation symmetry. The change in overall validation loss for a single gradient step with respect to $\boldsymbol{x}$ is:

$$\frac{1}{m_{\text{val}}} \sum_{i=1}^{m_{\text{val}}} \Big(\ell(\boldsymbol{\theta}, \boldsymbol{z}_i) - \ell(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{z}_i) \Big) \approx \frac{1}{m_{\text{val}}} \sum_{i=1}^{m_{\text{val}}} \eta \langle \nabla \ell(\boldsymbol{\theta}, \boldsymbol{z}_i), \nabla \ell(\boldsymbol{\theta}, \boldsymbol{x}) \rangle. \tag{5}$$

On the other hand, performing a batch gradient step at $\boldsymbol{\theta}$ with respect to the validation set gives $\boldsymbol{\theta}_{\boldsymbol{Z}^{\text{val}}} = \boldsymbol{\theta} - \eta \frac{1}{m_{\text{val}}} \sum_{i=1}^{m_{\text{val}}} \nabla \ell(\boldsymbol{\theta}, \boldsymbol{z}_i)$. Combining this with Eq. (5), we get

$$\frac{1}{m_{\text{val}}} \sum_{i=1}^{m_{\text{val}}} \Big(\ell(\boldsymbol{\theta}, \boldsymbol{z}_i) - \ell(\boldsymbol{\theta}_{\boldsymbol{x}}, \boldsymbol{z}_i) \Big) \approx \langle \boldsymbol{\theta} - \boldsymbol{\theta}_{\boldsymbol{Z}^{\text{val}}}, \nabla \ell(\boldsymbol{\theta}, \boldsymbol{x}) \rangle \approx \ell(\boldsymbol{\theta}, \boldsymbol{x}) - \ell(\boldsymbol{\theta}_{\boldsymbol{Z}^{\text{val}}}, \boldsymbol{x})\,. \tag{6}$$

In other words, the change in average validation loss from training on $\boldsymbol{x}$ can be approximated by the change in loss on $\boldsymbol{x}$ after training on the validation set $\boldsymbol{Z}^{\text{val}}$.

Our main objective is to evaluate the left-hand side of Eq. (6) for all $\boldsymbol{x}$ in the training set. The right-hand side provides a far more efficient route: (i) Compute the loss $\ell(\boldsymbol{\theta}, \boldsymbol{x})$ for all training examples; (ii) fine-tune $\boldsymbol{\theta}$ on the validation set to obtain $\boldsymbol{\theta}_{\boldsymbol{Z}^{\text{val}}}$; (iii) re-evaluate the new loss $\ell(\boldsymbol{\theta}_{\boldsymbol{Z}^{\text{val}}}, \boldsymbol{x})$ at each training sample $\boldsymbol{x}$, and approximate the effect of training on $\boldsymbol{x}$ by computing the difference with the loss at point (i).

This requires one epoch of training on the validation set and two evaluations over the training pool, as opposed to N evaluations of the validation loss as suggested by a direct evaluation of the left-hand side of Eq. (6), and it does not require access to per-example gradients.

In the next sections we use this idea to obtain a selection algorithm that alternates training on a subset of the training set and on the validation set. A specific implementation, which we refer to as 'Method A', is given in Algorithm 1; a slightly different implementation ('Method B') will be given in Algorithm 2. In Method A, we start with a small random subset $U \subset [N]$ of the training pool. We train on U for L epochs, resulting in models $\hat{\boldsymbol{\theta}}_1^{\text{bas}}, \dots \hat{\boldsymbol{\theta}}_L^{\text{bas}}$. For each epoch $k \in [L]$ we fine-tune $\hat{\boldsymbol{\theta}}_k^{\text{bas}}$ for one epoch on the validation set, resulting in models $\hat{\boldsymbol{\theta}}_k^{\text{val}}$. For each epoch, every remaining

Algorithm 1 ToV Scoring Algorithm: *Method A.*

```
Input: Pretrained model $\boldsymbol{\theta}_0$, validation set $\boldsymbol{Z}^{\text{val}}$, training pool $\boldsymbol{X} = (\boldsymbol{x}_i : i \in [N])$, epochs $L$,
    selected data count $n$, learning-rate schedule $\{\eta_k\}_{k=1}^{L}$, base model count $m$, $\varepsilon \in [0,1)$
Output: Set of examples $S \subset [N]$ of size $n$
Sample base subset $U \subseteq [N]$ of size $m$ randomly; define $\boldsymbol{X}_U = (\boldsymbol{x}_i : i \in U)$
Initialize model: $\hat{\boldsymbol{\theta}}_0^{\text{bas}} \leftarrow \boldsymbol{\theta}_0$; set scores $\phi_i \leftarrow 0$ for all $i \in [N] \setminus U$
for $k = 1$ to $L$ do
    Train $\hat{\boldsymbol{\theta}}_{k-1}^{\text{bas}}$ on $\boldsymbol{X}_U$ for one epoch with learning rate $\eta_k$ to obtain $\hat{\boldsymbol{\theta}}_k^{\text{bas}}$
    Train $\hat{\boldsymbol{\theta}}_k^{\text{bas}}$ for one epoch on $\boldsymbol{Z}^{\text{val}}$ with a learning rate $\varepsilon\eta_k$ to obtain $\hat{\boldsymbol{\theta}}_k^{\text{val}}$
    for each $i \in [N] \setminus U$ do
        $\phi_i^{(k)} \leftarrow F(\ell(\hat{\boldsymbol{\theta}}_k^{\text{val}}; \boldsymbol{x}_i) - \ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}}; \boldsymbol{x}_i))$ (see Section 2.1 for the definition of $F$)
        $\phi_i \leftarrow \phi_i + \phi_i^{(k)}/L$
    end for
end for
Return set $S \subseteq [N] \setminus U$ of size $n$ on the basis of scores $\phi_i$ (see text)
```

training example $\boldsymbol{x}_i$ with $i \in [N] \setminus U$ is scored by the change in its loss between $\hat{\boldsymbol{\theta}}_k^{\text{bas}}$ and $\hat{\boldsymbol{\theta}}_k^{\text{val}}$, and scores are averaged across epochs.

After computing scores ϕ_i as in Algorithm 1, we select S using one of two strategies: (i) choose the n examples with the largest ϕ_i; (ii) choose half from the highest-scoring examples and the other half uniformly at random from U to increase diversity.

Intuitively, large ϕ_i means that a small amount of training on the target distribution produces a large change in the model output at $\boldsymbol{x}_i$. Our working assumption, motivated by the heuristics above and formalized in Section 3, is that the converse also holds: training on $\boldsymbol{x}_i$ will produce a large change in model output on the target distribution. Hence the scores ϕ_i can be used to select 'important' samples for the target.

An adaptation for token-based learning is described in Section 2, along with empirical results. Section 3 provides a mathematical justification that formalizes the argument above.

1.2 Related work

Our work relates to data selection and data attribution. The impact of a single example on the validation error can be approximated by a first-order Taylor expansion. This idea results in data selection methods based on influence functions (Wang et al., 2018; 2020; Ai et al., 2021; Kolossov et al., 2024). Classical influence functions estimate the effect of a single example on the empirical risk minimizer. Most closely related to our work are Pruthi et al. (2020); Bae et al. (2024); Xia et al. (2024), which instead estimate the influence of an example on the training dynamics. In particular, Bae et al. (2024) shows how to approximately propagate gradient changes at k-th epoch through all subsequent epochs. In contrast, Pruthi et al. (2020); Xia et al. (2024) make a crude approximation for this propagation. Limitations of influence-based methods are discussed in Schioppa et al. (2023).

The recent work of Xia et al. (2024) proposes LESS, a data selection method for instruction tuning that adapts influence ideas to Adam and long sequences. In particular, these authors emphasize the challenge of computing and storing gradients to compute influences. They address this problem via random projections and low-rank approximation. Engstrom et al. (2024) apply the datamodel framework (Ilyas et al., 2022; Park et al., 2023) to select pretraining data. Separately, a *replay* algorithm that stores only a logarithmic number of checkpoints is proposed in Engstrom et al. (2025). Methods that align training data distributions to a small target set include TSDS (Liu et al., 2024) and DSIR (Xie et al., 2023); domain/task-adaptive pretraining also improves transfer (Gururangan et al., 2020). Broader LLM data-efficiency work proposes LLM-guided quality scoring (Ask-LLM) and density sampling (Sachdeva et al., 2024), and clustering-based sensitivity sampling with provable guarantees (Axiotis et al., 2024). Finally, Data Filtering Networks (DFN) also leverage a held-out, high-quality set, but with a different goal and setup (Fang et al., 2023).

Our contribution differs by (i) inverting train/validation roles to approximate per-example influence using only forward losses and doesn't require per example gradients, or Hessian-vector products—and (ii) showing that this simple, symmetry-based score is computationally inexpensive and outperforms recent data selection approaches for instruction tuning and NER.

2 Data selection for token-based learning

In this section we describe our implementation of the general idea described in the introduction for token-based learning and present empirical results demonstrating its effectiveness. Since prediction takes place at the token level, while data selection takes place at the example level (e.g., instruction/output pair), we compute token scores and aggregate them as described in Section 2.1. Section 2.2 gives a brief overview of instruction-tuning and NER tasks. Experimental settings are introduced in Section 2.3. Empirical results are presented in Sections 2.4 and 2.5.

2.1 Score computation for token-based learning

Each example $\boldsymbol{z}$ consists of an input $\boldsymbol{z}^{\text{in}}$ and an output $\boldsymbol{z}^{\text{out}}$, both of which are strings and may differ in length. Let $\mathcal{Z}^{\text{out}}$ denote the output vocabulary, and let $T(\boldsymbol{z})$ denote the length of the output string $\boldsymbol{z}^{\text{out}}$, which we write as $\boldsymbol{z}^{\text{out}} = \big(z^{\text{out}}(1), z^{\text{out}}(2), \dots, z^{\text{out}}(T(\boldsymbol{z}))\big)$.

Given a model parameterized by $\boldsymbol{\theta}$, its prediction on example $\boldsymbol{z}$ is a sequence of $T(\boldsymbol{z})$ conditional distributions, $\{p_t(\cdot \mid \boldsymbol{z}, \boldsymbol{\theta})\}_{t=1}^{T(\boldsymbol{z})}$, where each $p_t(\cdot \mid \boldsymbol{z}, \boldsymbol{\theta})$ denotes the model's predictive distribution over the output token at position t. Note that $p_t(\cdot \mid \boldsymbol{z}, \boldsymbol{\theta})$ depends on $\boldsymbol{z}$ solely through $\boldsymbol{z}^{\text{in}}$ and $z^{\text{out}}(1), \dots, z^{\text{out}}(t-1)$. We train models using the log-loss

$$\ell(\boldsymbol{\theta}; \boldsymbol{z}) = -\tfrac{1}{T(\boldsymbol{z})} \textstyle\sum_{t=1}^{T(\boldsymbol{z})} \log p_t\big(z^{\text{out}}(t) \mid \boldsymbol{z}; \boldsymbol{\theta}\big). \tag{7}$$

To compare two models, $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, on example $\boldsymbol{z}$, we define a per-token difference of log-loss

$$\Delta_t(\boldsymbol{z}; \boldsymbol{\theta}, \boldsymbol{\theta}') = \log \frac{p_t\big(z^{\text{out}}(t) \mid \boldsymbol{z}; \boldsymbol{\theta}'\big)}{p_t\big(z^{\text{out}}(t) \mid \boldsymbol{z}; \boldsymbol{\theta}\big)}. \tag{8}$$

Since our setting involves selecting entire examples rather than individual tokens, we aggregate the per-token differences into a single score per example. Specifically, we apply a transformation function $F : \mathbb{R} \to \mathbb{R}$ to each Δ_t before averaging across positions. The final score for example $\boldsymbol{z}$ is:

$$\phi(\boldsymbol{z}; \boldsymbol{\theta}, \boldsymbol{\theta}') = \tfrac{1}{T(\boldsymbol{z})} \textstyle\sum_{t=1}^{T(\boldsymbol{z})} F\big(\Delta_t(\boldsymbol{z}; \boldsymbol{\theta}, \boldsymbol{\theta}')\big). \tag{9}$$

We consider three instantiations of the function F, leading to three different scoring methods:

Maximum-Improvement: $F(y) = y$ — emphasizes raw improvement.

Maximum-Absolute Change: $F(y) = |y|$ — captures the magnitude of change.

Maximum-Positive Improvement: $F(y) = \max\{y, 0\}$ — ignores degradations.

The algorithm is therefore the same as in Algorithm 1, with the adaptation $\phi_i^{(k)} = \phi(\boldsymbol{x}_i; \hat{\boldsymbol{\theta}}_k^{\text{bas}}, \hat{\boldsymbol{\theta}}_k^{\text{val}})$.

Given a budget of n examples, we choose $S \subseteq [N] \setminus U$, $|S| = n$ using one of these rules:

Score-only: Choose the n examples $i \in [N] \setminus U$ that have the largest score ϕ_i.

Score+random: Choose the $n/2$ examples $i \in [N] \setminus U$ that have the largest score ϕ_i, and add $n/2$ more examples chosen uniformly at random (without replacement) from U.

Our scoring schemes tend to favor shorter examples due to their higher variance, which arises from having fewer tokens. To mitigate this bias, we partition the set $[N] \setminus U$ into 10 bins based on sequence length, ensuring each bin contains an equal number of examples. We then select an equal number of top-scoring examples from each bin.

After selecting S of size $|S| = n$, we train (or fine tune) a model on S to evaluate the selection scheme. We refer to this stage as *final training*.

We compare our schemes with three baselines:

RANDOM: The set S is selected uniformly at random subject to its size.

MAXIMUM UNCERTAINTY: Instead of the scores we defined, we use the following hardness score:

$$\psi_i := \frac{1}{T_i}\sum_{t=1}^{T_i} \log\left(\mathsf{p}_t(z_i(t)|\boldsymbol{z}_i;\hat{\boldsymbol{\theta}}_L^{\text{bas}})(1-\mathsf{p}_t(z_i(t)|\boldsymbol{z}_i;\hat{\boldsymbol{\theta}}_L^{\text{bas}})\right), \tag{10}$$

This score extends the method of Ting & Brochu (2018); Wang et al. (2018); Ai et al. (2021); Kolossov et al. (2024) to token-based learning .

LESS: We used the publicly available implementation from Xia et al. (2024); see Appendix A.1.

2.2 PREDICTION TASKS

We evaluate our data selection framework in two distinct token-based tasks: instruction tuning (IT) and named entity recognition (NER). The framework we introduced above captures both tasks:

Instruction Tuning (IT) involves training a language model to follow natural language instructions. Each training example consists of:
Input $\boldsymbol{z}^{\text{in}}$: a user instruction or prompt; **Output $\boldsymbol{z}^{\text{out}}$**: the desired model response.

The output is typically multi-token and highly variable in content and length, depending on the instruction. The model learns to generate $\boldsymbol{z}^{\text{out}}$ conditioned on $\boldsymbol{z}^{\text{in}}$. This naturally fits our framework, which models predictions as token-level distributions $p_t(\cdot \mid \boldsymbol{z}, \boldsymbol{\theta})$.

Named Entity Recognition (NER) is a sequence labeling task where the model assigns a probability distribution over entity tags (e.g., PERSON, ORGANIZATION, . . .) to each token. In this case:
Input $\boldsymbol{z}^{\text{in}}$: a tokenized sentence; **Output $\boldsymbol{z}^{\text{out}}$**: a sequence of entity labels, aligned with the input.

In NER, predictions are computed as token-wise classification distributions and therefore output is of the same length as input sequence.[1] In this case, as a base model we take a pretrained language model and replace its prediction head with a classification head.

2.3 EXPERIMENTAL SETTING

In all of our experiments the training set consisted of $N = 36 \times 1024$ samples. For the base model training, we used $|U| = 4 \times 1024$ samples. The validation set size is $m_{\text{val}} = 1024$ and the test set size is $m_{\text{tst}} = 10,000$. We vary the selected set size $n \in \{1, 2, 4, 8\} \times 1024$.

Number of epochs. Both for surrogate model training and final model training we determine the number of epochs by $L = (16 \times 1024)/n_{\text{tr}}$. We use a batch size of 16 whence the above ensures that the number batches used in training remains constant, and equal to 1024. In other words, all experiments in this section are at *constant compute*. Since base model training uses $|U| = m = 4 \times 1024$ samples, the number of epochs is $L = 4$.

Learning rate. The learning rate for both surrogate and final model training is selected using hyperparameter optimization for each selected set size n. The learning-rate optimization was carried out for random data selection hence placing our approach at a disadvantage.

We use linear learning rate scheduler and LoRA training Hu et al. (2022) with LoRA parameters $\alpha = 32$ and dropout $= 0.2$. For NER experiments, we used $\mathsf{PEFTrank} = 1$ and for instruction tuning experiments, we used $\mathsf{PEFTrank} = 256$. The learning rate for the validation examples is $\varepsilon = 1/10$ of the one for the base examples. We present here results with SCORE+RANDOM and refer to the appendix for SCORE-ONLY.

2.4 EXPERIMENTS FOR INSTRUCTION TUNING

For these experiments we used 3 different datasets, which we will refer to as $\mathscr{S} :=$ {Slim Orca, Alpaca GPT-4, Alpaca GPT-3.5}. As the foundation model, we use **Meta-Llama-3-8B**. Additional details of the model and datasets used are provided in the Appendix.

[1] In NER, typically token level probabilities are combined to assign labels to a whole word.

We designed five experimental setups. In each experiment, one dataset from $\mathscr{S}$ is selected as the *target distribution*. We randomly sample validation and test sets, $\boldsymbol{Z}^{\text{val}}$ and $\boldsymbol{Z}^{\text{tst}}$, without replacement from the target dataset. These samples are excluded from further use. The *training pool* is then formed by randomly sampling an equal number of examples from one or more datasets in $\mathscr{S}$ (excluding the validation and test samples), such that the total number of selected training samples is fixed at N. We denote by $\mathscr{S}_* \subseteq \mathscr{S}$ the datasets used to generate the training pool. The choices of the target dataset and of $\mathscr{S}_*$ for each of the five experiments are summarized in Table 1. All reported results are averaged over 10 independent runs. In each run, we freshly sample the training, validation, and test sets. These experiments are designed to evaluate performance across a range of data configurations. In particular: in Experiments 1 and 4, the training set includes samples from both target distribution and other distributions; in Experiments 2 and 5, the training set includes samples only from non-target distributions; in Experiment 3, it includes only samples from the target distribution.

Table 1: Summary of instruction tuning experiments. Abbreviations: SO = Slim Orca, A4 = Alpaca GPT-4, A3.5 = Alpaca GPT-3.5.

Exp	Target	Training pool
1	SO	SO, A4, and A3.5
2	SO	A4 and A3.5
3	SO	SO
4	A4	SO, A4, and A3.5
5	A4	SO and A3.5

Table 2: Summary of named entity recognition experiments. Abbreviations: MN = Multinerd, A4p = Ai4p, C4 = C4, SB = Syn-big.

Exp	Target	Training pool
1	MN	MN, A4p, C4, and SB
2	MN	A4p, C4, and SB
3	MN	MN
4	A4p	MN, A4p, C4, and SB
5	A4p	MN, C4, and SB
6	A4p	A4p

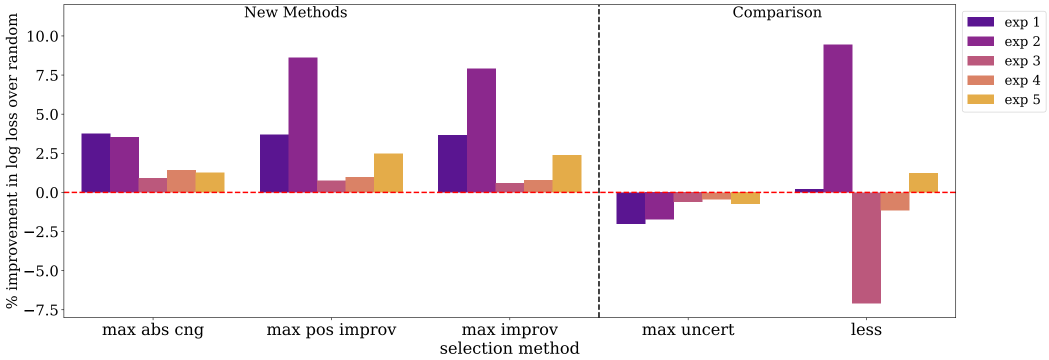

Figure 1: Test log-loss improvement (%) over random selection for instruction tuning with $n = 8 \times 1024$ samples. Each group of bars represents a data-selection strategy (maximum-uncertainty and LESS as baselines); colors show target/training pool configuration (Table 1). Results use Method A (Algorithm 1) with the SCORE+RANDOM strategy.

Figure 1 summarizes our results for instruction tuning for a fixed select size $n = 8 \times 1024$. We plot the improvement in test log-loss over random data selection for several data-selection strategies within the general framework described in Section 2.1, using method A in algorithm 1 for scoring the examples and SCORE+RANDOM for selecting. We observe that the proposed strategies yield significantly better instruction tuning than random data selection or selecting by max-uncertainty. We observe an improvement (albeit a small one) even when both train and validation data are from Slim Orca (Exp 3), which is a case in which random selection should perform well. The proposed strategies also yield a significant improvement over LESS (Xia et al., 2024), with the exception of Experiment 2 in which LESS performs slightly better.

Figure 2 displays the evolution of test log loss with selected sample size n. We observe that a good choice of the data selection method results in model improvements that can be equivalent to or larger than doubling n. Plots show standard error (with scaling factor 1) for 10 runs.

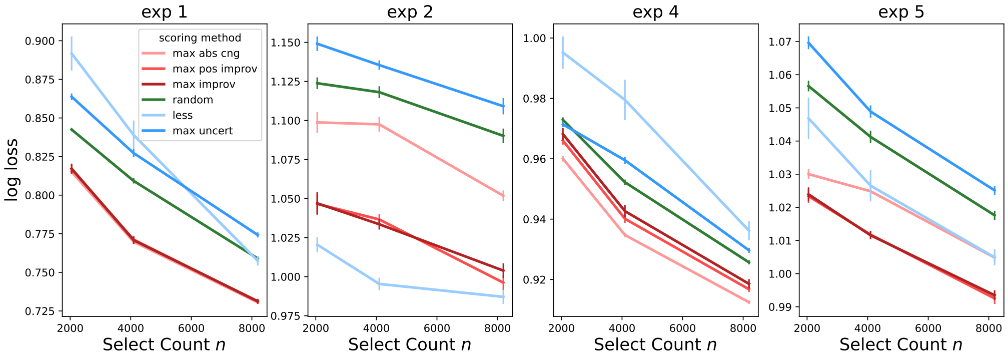

Figure 2: Test log-loss vs. number of selected samples n for instruction tuning. (Due to space limits, Exp. 3 plot is in the Appendix.) Lines show mean log-loss over 10 runs; error bars are ±1 standard error. Results use Method A with the SCORE+RANDOM strategy.

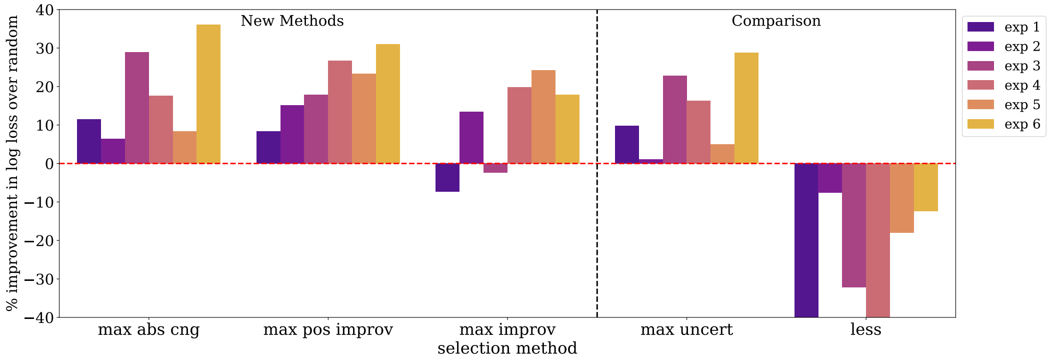

Figure 3: Test log-loss improvement (%) relative to random selection for NER at $n = 8 \times 1024$. Each group of bars represents a data-selection strategy; colors show target/training pool configuration (Table 2). Results use Method A (Algorithm 1) with the SCORE+RANDOM strategy

2.5 Experiments for named entity recognition

The task is to classify whether a token is part of a person name or not. For these experiments we used 4 different labeled datasets, which we will refer to as $\mathscr{S} := \{$ Multinerd, Ai4p, C4, Syn-big$\}$. We use **xlm-roberta-base** as the foundation model. Further details on the experiment, model and datasets used are presented in the Appendix.

We conducted six sets of experiments. As for the case of instruction tuning, for each set of experiments, we select one of the datasets $\mathscr{S}$ as defining the target distribution, and one or more other datasets to define the training pool (denoted by $\mathscr{S}_*$). The choices of target datasets and $\mathscr{S}_*$ are summarized in Table 2. The construction of train, test and validation sets is same as in instruction tuning.

Figure 3 summarizes our experiments with NER. We plot the improvement in test log-loss over random data selection for several scores definitions. Throughout these experiments, we use SCORE+RANDOM. We observe that the strategies of Section 2.1 yield systematic improvements over random data-selection. Unlike in the case of instruction tuning, maximum uncertainty also yields an improvement in most settings. However, the ToV approach achieves a larger improvement. Finally, in this case LESS (Xia et al., 2024) appears not to improve over random data selection.

3 A formal justification

In this section we present a mathematical analysis of our approach in the case of batch gradient descent (GD). We focus on the implementation *Method B*, described in Algorithm 2.

Algorithm 2 ToV Scoring Algorithm: *Method B*

```
Input: Pretrained model $\boldsymbol{\theta}_0$, validation set $\boldsymbol{Z}^{\text{val}}$, training pool $\boldsymbol{X} = (\boldsymbol{x}_i : i \in [N])$,
        selected data count $n \leq N$, base model count $m$
Output: Set of examples $S \subset [N]$ of size $n$
Sample base subset $U \subseteq [N]$ of size $m$ randomly; define $\boldsymbol{X}_U = (\boldsymbol{x}_i : i \in U)$
Initialize models: $\hat{\boldsymbol{\theta}}_0^{\text{bas},+} \leftarrow \boldsymbol{\theta}_0$, $\hat{\boldsymbol{\theta}}_0^{\text{bas}} \leftarrow \boldsymbol{\theta}_0$; set scores $\Upsilon_i \leftarrow 0$ for all $i \in [N] \setminus U$
for $k = 1$ to $L$ do
    Train for one epoch on $\boldsymbol{X}_U$ with learn. rate $\eta_k$ and init. $\hat{\boldsymbol{\theta}}_{k-1}^{\text{bas},+}$. Denote the output by $\hat{\boldsymbol{\theta}}_{0,k}^{\text{bas},+}$
    Train for one epoch on $\boldsymbol{Z}^{\text{val}}$ with learn. rate $\varepsilon \cdot \eta_k$ and init. $\hat{\boldsymbol{\theta}}_{0,k}^{\text{bas},+}$. Denote the output by $\hat{\boldsymbol{\theta}}_k^{\text{bas},+}$
    Train for one epoch on $\boldsymbol{X}_U$ with learn. rate $\eta_k$ and init. $\hat{\boldsymbol{\theta}}_{k-1}^{\text{bas}}$. Denote the output by $\hat{\boldsymbol{\theta}}_k^{\text{bas}}$
    for each $i \in [N] \setminus U$ do
        $\Upsilon_i^{(k)} \leftarrow \ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}}; \boldsymbol{x}_i) - \ell(\hat{\boldsymbol{\theta}}_k^{\text{bas},+}; \boldsymbol{x}_i)$
        $\Upsilon_i \leftarrow \Upsilon_i + \Upsilon_i^{(k)}/L$
    end for
end for
Select $S \subseteq [N] \setminus U$ with size $|S| = n$ using scores $\Upsilon_i$
```

Method B differs from Method A because at each training cycle k, training on the base set $\boldsymbol{X}_U$ is initialized with the output of the previous train-on-validation phase. Empirically Method A performs somewhat better than B, see Appendix C. We use Method B for analysis just because the resulting mathematical expressions are simpler.

We find empirically that the ToV works well beyond token-based learning, and hence our focus will be to understand it in a generic learning problem. Appendix D demonstrates this point by considering a simple logistic regression problem.

3.1 Ideal scores, linearization, influence functions

In order to estimate the model improvement produced by sample $i \in [N] \setminus U$ we could train a model on two training sets $\boldsymbol{X}_U$ and $\boldsymbol{X}_{U \cup i}$, using empirical risk functions $\widehat{R}_U(\boldsymbol{\theta})$, $\widehat{R}_{U \cup i}(\boldsymbol{\theta})$. We thus would run GD for L steps, with initialization $\hat{\boldsymbol{\theta}}_0^{\text{bas}} = \hat{\boldsymbol{\theta}}_0^{\text{bas}+i} = \boldsymbol{\theta}_0$:

$$\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}} = \hat{\boldsymbol{\theta}}_k^{\text{bas}} - \eta m \nabla \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}}) , \quad \hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}+i} = \hat{\boldsymbol{\theta}}_k^{\text{bas}+i} - \eta(m+1) \nabla \widehat{R}_{U \cup i}(\hat{\boldsymbol{\theta}}_k^{\text{bas}+i}) . \tag{11}$$

At iteration k, we have thus two models $\hat{\boldsymbol{\theta}}_k^{\text{bas}}$ and $\hat{\boldsymbol{\theta}}_k^{\text{bas}+i}$ that differ uniquely in whether sample i is used or not. We define the *ideal score* to be the difference in validation error between these two models, averaged over epochs

$$S_i := \frac{1}{L} \sum_{s=1}^{L} [\widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_s^{\text{bas}}) - \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_s^{\text{bas}+i})] = \frac{1}{m_{\text{val}} L} \sum_{s=1}^{L} \sum_{j=1}^{m_{\text{val}}} \left\{ \ell(\hat{\boldsymbol{\theta}}_s^{\text{bas}}; \boldsymbol{z}_j^{\text{val}}) - \ell(\hat{\boldsymbol{\theta}}_s^{\text{bas}+i}; \boldsymbol{z}_j^{\text{val}}) \right\} . \tag{12}$$

Evaluating this score is computationally expensive, hence several groups (Pruthi et al., 2020; Bae et al., 2024; Xia et al., 2024) proposed to use a first order Taylor expansion to approximate the difference between the two models. Expanding S_i with respect to the contribution of $\ell(\,\cdot\,; \boldsymbol{x}_i)$ yields

$$S_i^{\text{lin}} = \tfrac{\eta}{L} \textstyle\sum_{0 \leq s < t \leq L} \langle \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_t^{\text{bas}}), \boldsymbol{M}_{t,s+1} \nabla \ell(\hat{\boldsymbol{\theta}}_s^{\text{bas}}; \boldsymbol{x}_i) \rangle . \tag{13}$$

where $\boldsymbol{M}_{t,t} = \boldsymbol{I}_d$ and $\boldsymbol{M}_{t,r}$ captures the propagation of perturbations along the GD trajectory:

$$\boldsymbol{M}_{t,r} := \boldsymbol{H}_{t-1} \cdot \boldsymbol{H}_{t-2} \cdots \boldsymbol{H}_r , \qquad \boldsymbol{H}_k := \boldsymbol{I} - \eta m \nabla^2 \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}}) . \tag{14}$$

The next result shows that S_i^{lin} approximates well S_i in a quantitative way, under local convexity.

Proposition 1. *Assume there exist $c_0, C_1, M > 0$ such that $\nabla^2 \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}}) \succeq c_0 \boldsymbol{I}_d$, $\|\nabla \ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}}; \boldsymbol{x}_i)\| \leq C_1$ for all k and, for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, $\|\nabla^2 \widehat{R}_U(\boldsymbol{\theta}_1) - \nabla^2 \widehat{R}_U(\boldsymbol{\theta}_2)\|_{\text{op}} \leq M \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2$, $\|\nabla \ell(\boldsymbol{\theta}_1; \boldsymbol{x}_i) - \nabla \ell(\boldsymbol{\theta}_2; \boldsymbol{x}_i)\|_{\text{op}} \leq M \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2$. Further assume that $\|\nabla^2 \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\|_{\text{op}} \leq C_1$ and $\|\nabla^2 \widehat{R}_{\text{val}}(\boldsymbol{\theta}_1) -$*

$\nabla^2 \widehat{R}_{\text{val}}(\boldsymbol{\theta}_2)\|_{\text{op}} \le M\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2$ *for all* $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ *as well. Finally, assume there exists a constant* C_η *such that* $\eta_k = \eta \le C_\eta/m \ \forall k$. *Then there exists* $C = C(c_0, C_1, C_\eta, M)$ *such that*

$$\big|S_i - S_i^{\text{lin}}\big| \le C/m^2 \,. \tag{15}$$

The assumption $\eta \le C_\eta/m$ is justified by the fact that we expect the Hessian of $\widehat{R}_U(\,\cdot\,)$ to be of order one, and hence the stepsize for this objective (which is given by ηm see Eq. (11)) should be of order one. As shown in the proof, the typical size of S_i^{lin} is of order $1/m$, and hence Eq. (15) establishes that the difference $|S_i - S_i^{\text{lin}}|$ is negligible.

3.2 Train-validation duality

We consider Methods A and B defined in Algorithms 1, 2. We emphasize the dependence on ε by writing $\phi_i = \phi_i(\varepsilon)$ and $\Upsilon_i = \Upsilon_i(\varepsilon)$. It is easy to derive the small ε asymptotics $\phi_i = \phi_i^{\text{lin}}\varepsilon + o(\varepsilon)$, $\Upsilon_i = \Upsilon_i^{\text{lin}}\varepsilon + o(\varepsilon)$, where, for $\boldsymbol{g}_{s,i} := \nabla\ell(\hat{\boldsymbol{\theta}}_s^{\text{bas}}; \boldsymbol{x}_i)$,

$$\phi_i^{\text{lin}} := \frac{\eta m_{\text{val}}}{L}\sum_{s=1}^{L}\langle \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_s^{\text{bas}}), \boldsymbol{g}_{s,i}\rangle\,, \qquad \Upsilon_i^{\text{lin}} := \frac{\eta m_{\text{val}}}{L}\sum_{0\le t<s\le L}\langle \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_{t+1}^{\text{bas}}), \boldsymbol{M}_{s,t+1}^{\mathsf{T}}\boldsymbol{g}_{s,i}\rangle\,. \tag{16}$$

We show that these are good approximations of $\Upsilon_i(\varepsilon), \phi_i(\varepsilon)$ uniformly in dimension, sample size.

Theorem 1. *Consider Algorithms 1, 2 with fixed stepsize* $\eta_k = \eta$ *(and* $F(x) = -x$ *in Algorithm 1). Under the assumptions of Proposition 1, further assume* $\|\nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\| \le C_1$ *for all* k, *and* $\|\nabla^2_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta}; \boldsymbol{x})\|_{\text{op}} \le C_1$. *Then there exist* $c_* = c_*(c_0, M, C_1)$, $C = C(c_0, M, C_1)$ *such that, for* $\varepsilon m_{\text{val}}/m \le c_*$,

$$\big|\Upsilon_i(\varepsilon) - \Upsilon_i^{\text{lin}}\varepsilon\big| \le C\big(\varepsilon m_{\text{val}}/m\big)^2\,, \quad \big|\phi_i(\varepsilon) - \phi_i^{\text{lin}}\varepsilon\big| \le C\big(\varepsilon m_{\text{val}}/m\big)^2\,. \tag{17}$$

Note that Υ_i^{lin} differ from S_i^{lin}. because of: (i) The different order of s and t; (ii) The fact that $\boldsymbol{M}_{t,s+1}$ is replaced by its transpose in Eq. (16). Υ_i^{lin} measures the influence of training on validation data when making inference at $\boldsymbol{x}_i$, while S_i^{lin} measures the influence of training on $\boldsymbol{x}_i$ data when making inference on validation. These two measures of 'influence' differ by the replacement of $\boldsymbol{M}_{t,s+1}$ by $\boldsymbol{M}_{s,t}^{\mathsf{T}}$. However, in a number of cases we expect these two matrices to be not too different, and hence the two scores to yield similar results. We can prove that Υ_i^{lin} and S_i^{lin} coincide (for large L) under local convexity conditions.

Theorem 2. *Assume* $\boldsymbol{\theta} \mapsto \ell(\boldsymbol{\theta}; \boldsymbol{x})$ *to be twice continuously differentiable and that* $\|\nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\| \le C_1$, $\|\nabla\ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}}; \boldsymbol{x}_i)\| \le C_1$ *for all* k. *Further assume that gradient descent iterates* $(\hat{\boldsymbol{\theta}}_k^{\text{bas}} : k \ge 0)$ *converge to* $\hat{\boldsymbol{\theta}}_\infty^{\text{bas}} = \lim_{k\to\infty}\hat{\boldsymbol{\theta}}_k^{\text{bas}}$ *which is a local minimum of* $\widehat{R}_U(\boldsymbol{\theta})$ *with* $\boldsymbol{Q}_\infty := \nabla^2\widehat{R}_U(\hat{\boldsymbol{\theta}}_\infty^{\text{bas}}) \succ \boldsymbol{0}$ *(strictly). Then*

$$\lim_{L\to\infty}\frac{1}{m_{\text{val}}}\Upsilon_i^{\text{lin}}(L) = \lim_{L\to\infty} S_i^{\text{lin}}(L) = \frac{1}{m}\langle \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_\infty^{\text{bas}}), \boldsymbol{Q}_\infty^{-1}\nabla\ell(\hat{\boldsymbol{\theta}}_\infty^{\text{bas}}; \boldsymbol{x}_i)\rangle := S_{i,\infty}^{\text{lin}}\,. \tag{18}$$

The last expression in Eq. (18) (denoted by $S_{i,\infty}^{\text{lin}}$) is the classical formula for influence functions of M-estimators (van der Vaart, 2000). Both our approach and the dynamical influence function $S_i^{\text{lin}}(L)$ can be regarded as approximations of $S_{i,\infty}^{\text{lin}}$ in this case.

In fine tuning, the model is likely to be overparametrized, and it is unrealistic to assume convergence to a strict minimum (with $\nabla^2\widehat{R}_U(\hat{\boldsymbol{\theta}}_\infty^{\text{bas}}) \succ \boldsymbol{0}$). On the other hand, the weights will not change significantly during this phase and it is reasonable to approximate fine-tuning as fitting an overparametrized linear model with respect to the empirical neural tangent features learnt in the pre-training phase.

Theorem 3. *Consider the loss function* $\ell(\boldsymbol{\theta}; \boldsymbol{x}) = (y(\boldsymbol{x}) - \langle\boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\theta}\rangle)^2/2$ *for some response variables* $y(\boldsymbol{x})$, *and featurization map* $\boldsymbol{\psi} : \mathbb{R}^d \to \mathbb{R}^p$, $p > m$. *Let* $\boldsymbol{\Psi} \in \mathbb{R}^{|U|\times p}$ *be the matrix with rows* $(\boldsymbol{\psi}(\boldsymbol{x}_j) : j \in U)$, $\boldsymbol{\Psi}_{\text{val}} \in \mathbb{R}^{m_{\text{val}}\times p}$ *be the matrix with rows* $(\boldsymbol{\psi}(\boldsymbol{z}_j^{\text{val}}) : j \le m_{\text{val}})$, $\boldsymbol{P}_\Psi$ *the projector to the kernel of* $\boldsymbol{\Psi}$, $\boldsymbol{y} = (y(\boldsymbol{x}_j) : j \in U)$, $\hat{\boldsymbol{\theta}} := \boldsymbol{\Psi}^\dagger\boldsymbol{y}$, $\boldsymbol{r}^{\text{val}} := (y(\boldsymbol{z}_j^{\text{val}}) - \langle\hat{\boldsymbol{\theta}}, \boldsymbol{\psi}(\boldsymbol{z}_j^{\text{val}})\rangle : j \le m_{\text{val}})$, $r(i) := y(\boldsymbol{x}_i) - \langle\hat{\boldsymbol{\theta}}, \boldsymbol{\psi}(\boldsymbol{x}_i)\rangle$. *If GD is initialized with* $\boldsymbol{\theta}_0 = \boldsymbol{0}$, *and we use constant stepsize* $\eta < \|\boldsymbol{\Psi}\|_{op}^2/2$, *then*

$$\lim_{L\to\infty}\frac{1}{Lm_{\text{val}}}\Upsilon_i^{\text{lin}}(L) = \lim_{L\to\infty}\frac{1}{L}S_i^{\text{lin}}(L) = \frac{\eta}{2}\,r(i)\langle\boldsymbol{r}^{\text{val}}, \boldsymbol{\Psi}_{\text{val}}^{\mathsf{T}}\boldsymbol{P}_\Psi\boldsymbol{\psi}(\boldsymbol{x}_i)\rangle\,. \tag{19}$$

REFERENCES

Mingyao Ai, Jun Yu, Huiming Zhang, and HaiYing Wang. Optimal subsampling algorithms for big data regressions. *Statistica Sinica*, 31(2):749–772, 2021.

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Kyriakos Axiotis, Vincent Cohen-Addad, Monika Henzinger, Sammy Jerome, Vahab Mirrokni, David Saulpic, David Woodruff, and Michael Wunder. Data-efficient learning via clustering-based sensitivity sampling: Foundation models and beyond. *arXiv preprint arXiv:2402.17327*, 2024.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentation. *arXiv:2405.12186*, 2024.

Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *Acta numerica*, 30:87–201, 2021.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019. URL http://arxiv.org/abs/1911.02116.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*, 2024.

Logan Engstrom, Andrew Ilyas, Benjamin Chen, Axel Feldmann, William Moses, and Aleksander Madry. Optimizing ml training with metagradient descent. *arXiv preprint arXiv:2503.13751*, 2025.

Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. Data filtering networks. *arXiv preprint arXiv:2309.17425*, 2023.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Germain Kolossov, Andrea Montanari, and Pulkit Tandon. Towards a statistical theory of data selection under weak supervision. In *The Twelfth International Conference on Learning Representations*, 2024.

Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and ”Teknium”. Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification, 2023. URL https://https://huggingface.co/Open-Orca/SlimOrca.

Zifan Liu, Amin Karbasi, and Theodoros Rekatsinas. Tsds: Data selection for task-specific model finetuning. *Advances in Neural Information Processing Systems*, 37:10117–10147, 2024.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pp. 22631–22648. PMLR, 2023.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Noveen Sachdeva, Benjamin Coleman, Wang-Cheng Kang, Jianmo Ni, Lichan Hong, Ed H Chi, James Caverlee, Julian McAuley, and Derek Zhiyuan Cheng. How to train data-efficient llms. *arXiv preprint arXiv:2402.09668*, 2024.

Andrea Schioppa, Katja Filippova, Ivan Titov, and Polina Zablotskaia. Theoretical and practical perspectives on what influence functions do. *Advances in Neural Information Processing Systems*, 36:27560–27581, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Simone Tedeschi and Roberto Navigli. MultiNERD: A multilingual, multi-genre and fine-grained dataset for named entity recognition (and disambiguation). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 801–812, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.60. URL https://aclanthology.org/2022.findings-naacl.60.

Daniel Ting and Eric Brochu. Optimal subsampling with influence functions. *Advances in neural information processing systems*, 31, 2018.

Aaad W van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 2000.

HaiYing Wang, Rong Zhu, and Ping Ma. Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association*, 113(522):829–844, 2018.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786, 2023.

Zifeng Wang, Hong Zhu, Zhenhua Dong, Xiuqiang He, and Shao-Lun Huang. Less is better: Unweighted data subsampling via influence function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6340–6347, 2020.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 54104–54132, 2024.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

A Additional Experimental Details and Results

A.1 Experiments for Token-Based Learning

In these experiments, we used pretrained models as base models and constructed training, validation, and test sets from real-world datasets. Details of the datasets and models are provided in Section J.

For each training example count—both for surrogate model training (used for scoring) and final model training—we selected the learning rate from the following grid:

```
[3e-6, 1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2].
```

The optimal learning rate was determined by training models on randomly sampled subsets from the training pool and evaluating their test log-loss. For each learning rate, the loss was averaged over 10 runs, with a new random subset used in each run. The best-performing learning rate was selected separately for each experimental configuration listed in Table 1 and Table 2.

Implementation details for Less (Xia et al., 2024) We used the public implementation from the authors' GitHub repository. The projection dimension was set to 8192. Learning rate and other hyperparameters were tuned identically for all approaches. For both our method and LESS, the surrogate model used the same number of samples and was trained for four epochs, matching the settings in the LESS paper. Following the original LESS procedure, we selected the top-scoring examples.

A.2 Expanded Results for Instruction Tuning

In the main paper, we compared our scoring methods for the Score+Random strategy. Due to space constraints, Figure 2 omitted results for Experiment 3. In Figure 4, we provide an expanded version that includes results for Experiment 3 as well.

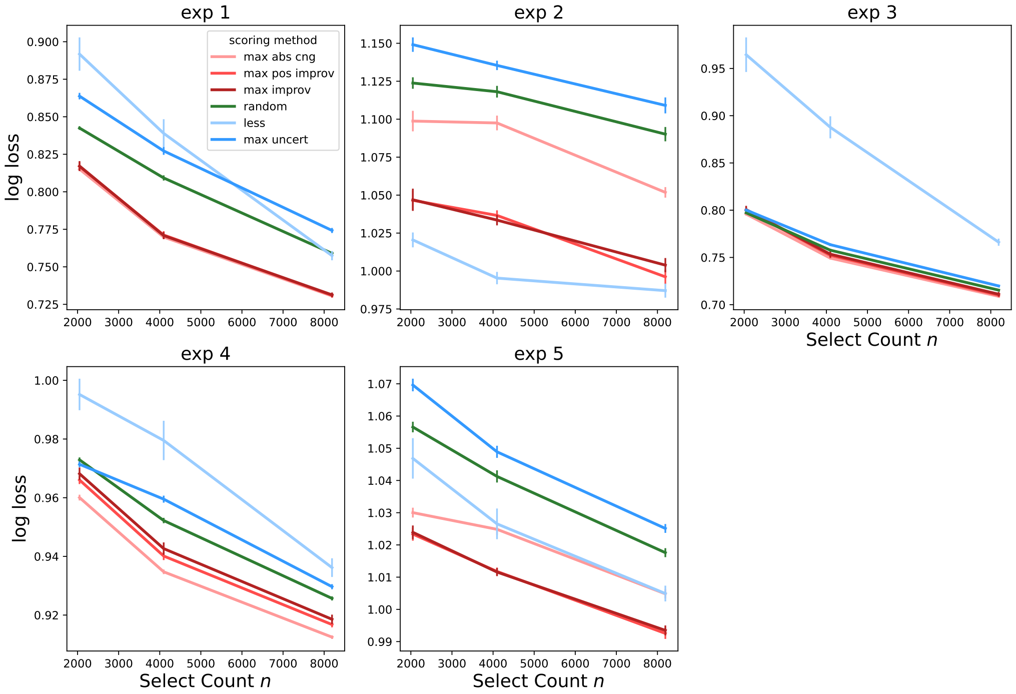

Figure 4: Expanded version of Figure 2 including the Experiment 3 plot.

A.3 Expanded Results for Named Entity Recognition

In Figure 3 of the main paper, we reported results for Score+Random using a fixed selected sample size of $n = 8 \times 1024$, across all experiment configurations in Table 2.

In Figure 5, we show how the test log-loss varies with the selected sample size n for different scoring methods under the Score+Random strategy, and how these compare to random selection.

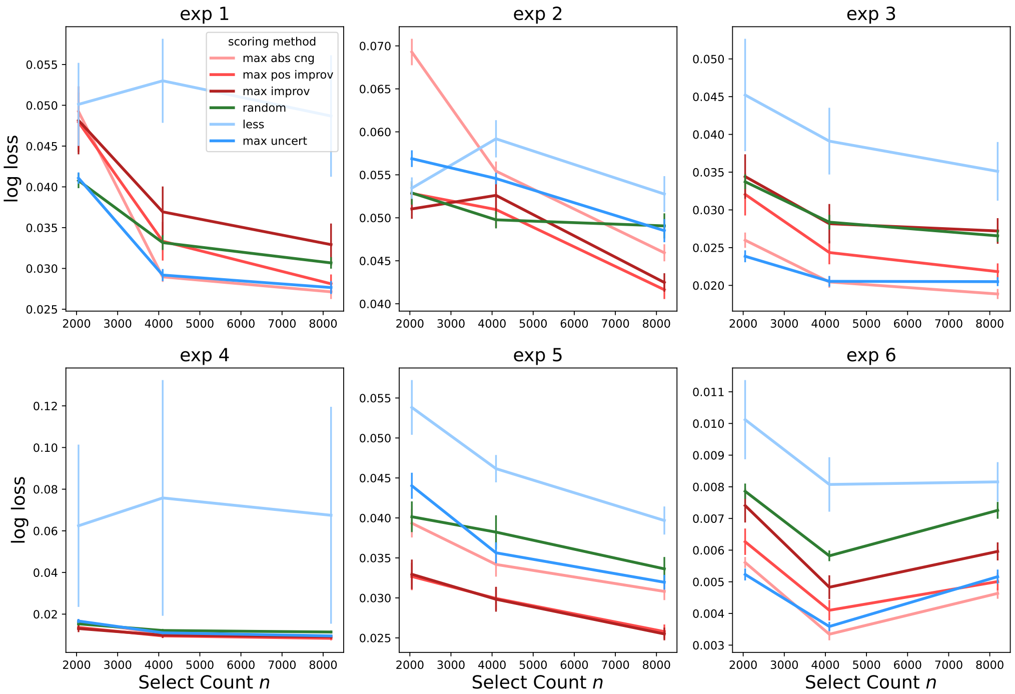

Figure 5: Test log-loss vs. number of selected samples n for NER. Lines show mean log-loss over 10 runs; error bars are ± 1 standard error. Results use Method A with the Score+Random strategy.

B Comparison of Score+Random and Score-Only selection

In this section we examine how the performance of our strategies changes when all training examples are selected from the top-scoring set (Score-Only) instead of selecting only half of them from the top and the other half at random (Score+Random).

Recall that our scores approximate how much benefit each example provides when added to a randomly chosen pool of training data. A higher score therefore indicates an example expected to be more helpful in that setting. Score+Random selects half of the final training set from the highest-scoring examples and fills the rest with random examples, whereas Score-Only takes only the top-scoring examples. This design creates a trade-off:

- Pure exploitation: Selecting only top-scoring examples can maximize immediate gain because every chosen example has a high estimated contribution.
- Score validity and diversity: The scores are defined relative to adding examples to a random pool. If we select only top examples, the resulting set may differ substantially from the random reference, making the scores a less accurate guide. Randomly adding half the examples keeps the final set closer to the conditions under which the scores were computed and also protects against loss of diversity.

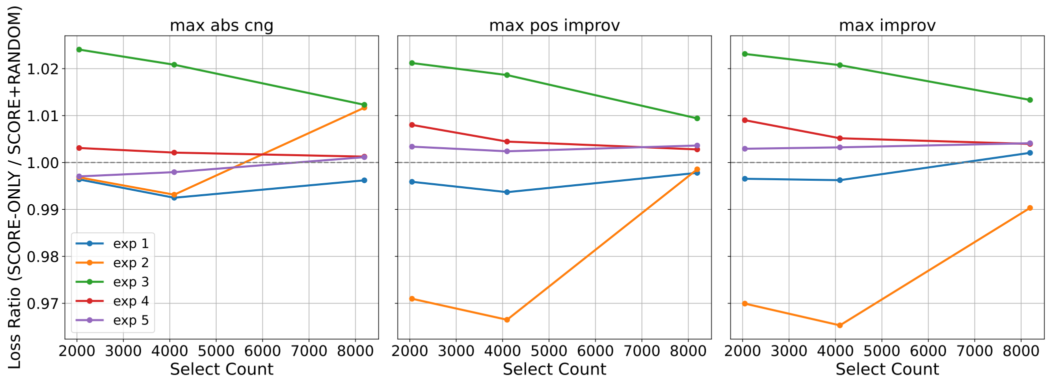

Figure 6: Ratio of log loss for Score-Only versus Score+Random across our three scoring strategies and all instruction-tuning setups in Table 1. Scores are computed using Algorithm 1.

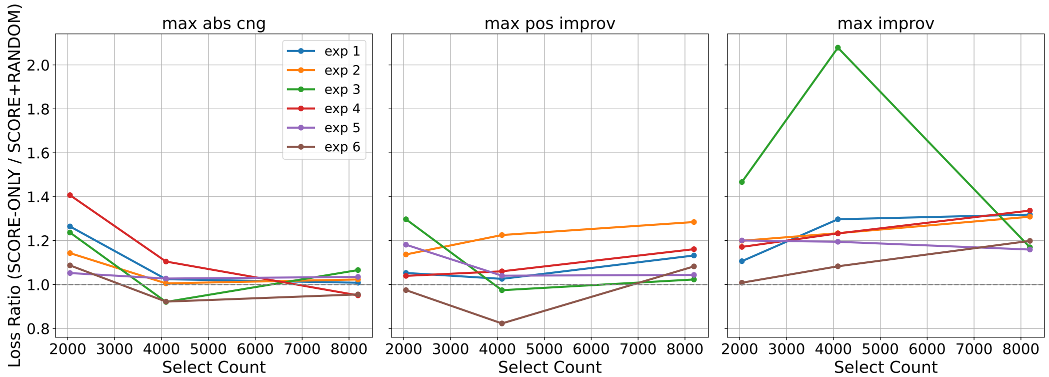

Figure 7: Ratio of log-loss for Score-Only versus Score+Random across our three scoring strategies and all NER setups in Table 2. Scores are computed using Algorithm 1.

Which effect dominates varies by task.

Figures 6 and 7 show the ratio of log-loss for the two selection strategies in instruction tuning and NER respectively. In each figure the three subplots correspond to our three scoring strategies; different lines indicate the various experimental setups. Algorithm 1 is used to obtain the scores.

For instruction tuning, Score+Random performs better in three setups (3, 4, 5), while Score-Only is better in the remaining two (1, 2) across most selection sizes and scoring methods. For NER, Score+Random tends to outperform Score-Only more often, particularly for the Max-Improvement scores.

C Method B vs Method A

All previous plots used Method A (Algorithm 1) for scoring. Here we compare the performance of the two scoring methods: Method A (Algorithm 1) and Method B (Algorithm 2)—across our experiments, using the Score+Random selection strategy for both.

Figures 8 and 9 show the ratio of test log-loss obtained with Method B relative to Method A for instruction tuning and NER, respectively. Each figure contains three subplots corresponding to our three scoring strategies, and different lines represent the various experimental setups.

The results indicate that for instruction tuning, Method A is most often superior, while for NER there is no consistent winner. A possible explanation is that Method B uses two distinct training trajectories. Our analysis assumes that the resulting models differ only slightly, but in practice, the two training trajectories can diverge substantially. This effect is likely to be stronger with large and highly overparameterized models such as Meta-Llama-3-8B, which we used for instruction tuning, We expect the larger distance between the two models to result in less accurate score estimation in Method B, as compared to Method A.

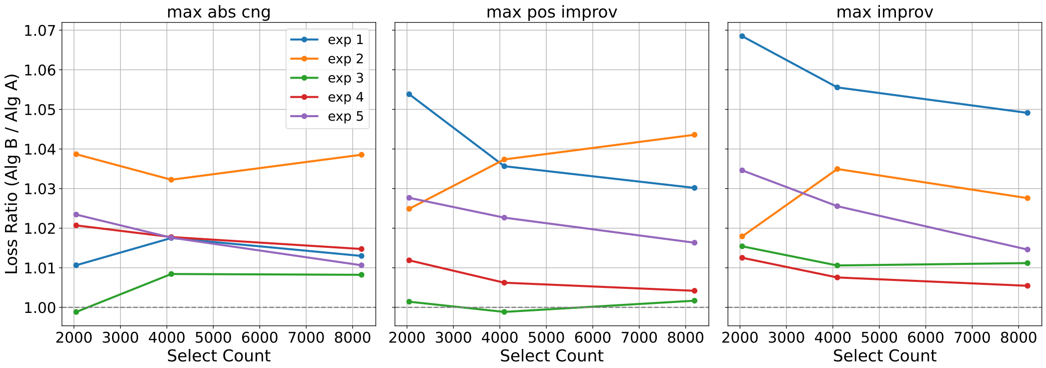

Figure 8: Ratio of test log-loss using Method B (Algorithm 2) to Method A (Algorithm 1) for instruction tuning. Results use the SCORE+RANDOM selection strategy. Each subplot corresponds to one scoring strategy; lines denote different experimental setups in Table 1.

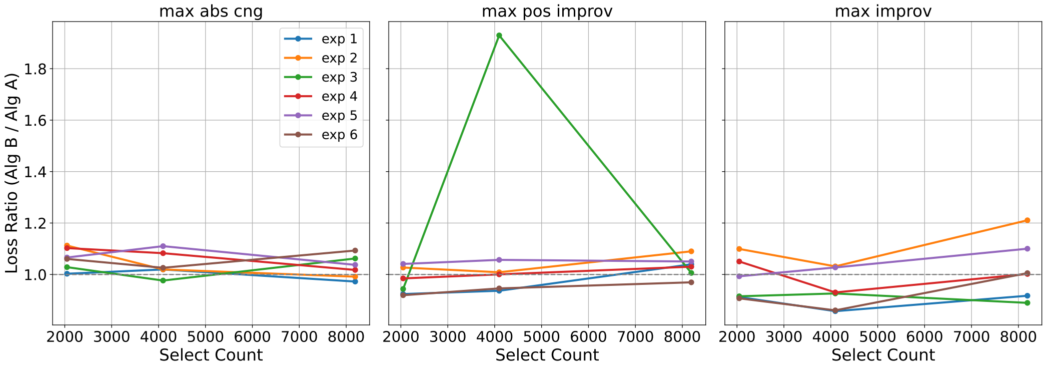

Figure 9: Ratio of test log-loss using Method B (Algorithm 2) to Method A (Algorithm 1) for NER. Results use the SCORE+RANDOM selection strategy. Each subplot corresponds to one scoring strategy; lines denote different experimental setups in Table 2.

D LOGISTIC REGRESSION EXPERIMENTS

In these experiments, we synthetically generated the training pool, validation set, and test set. We begin by defining a parametric family of distributions used to construct the data.

For a given $p > 0$ and parameter vector $\theta \in \mathbb{R}^p$, we define a distribution $\mathcal{P}_\theta$ over pairs (x, y), where $x \in \mathbb{R}^p$ and $y \in \{0, 1\}$. The features are sampled as $x \sim \mathcal{N}(0, I)$, and the label y is drawn according to a logistic model:

$$\Pr(y = 1 \mid x) = \frac{1}{1 + \exp(-x \cdot \theta)}, \quad \Pr(y = 0 \mid x) = 1 - \Pr(y = 1 \mid x).$$

We randomly sample a unit vector θ^* from the unit sphere to serve as the target direction. A second unit vector θ' is then drawn such that it lies at an angle γ from θ^*. In our experiments, we set $p = 10$ and $\gamma = \pi/2$.

The training pool consists of $N = 128 \times 1024$ samples, drawn independently from the mixture distribution:

$$\mathcal{D}_{\text{train}} = \frac{1}{2}\mathcal{P}_{\theta^*} + \frac{1}{2}\mathcal{P}_{\theta'}.$$

The validation and test sets contain $m_{\text{val}} = 1024$ and $m_{\text{tst}} = 10{,}000$ samples respectively, both drawn i.i.d. from the target distribution $\mathcal{P}_{\theta^*}$.

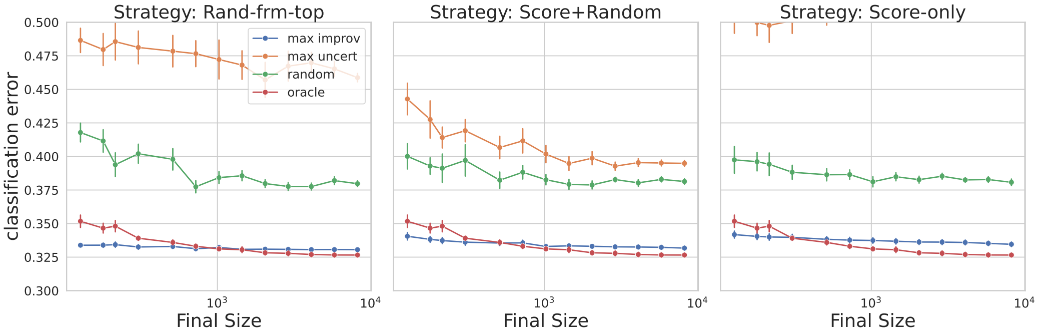

Figure 10: Data selection experiments with Method B for logistic regression on synthetic data in $d = 10$ dimensions (4 epochs of training). Each color corresponds to a distinct method to score data in the training pool, and each frame to a distinct method to use the score to form the selected set. Each symbol corresponds to the average of 10 experiments.

For scoring, we used Method B as described in Algorithm 2; similar experiments with Method A produced comparable results, so we report only Method B here. Algorithm 2 does not specify the method to select data on the basis of scores. In Figure 10 we compare SCORE-ONLY and SCORE+RANDOM (already introduced above) with a third one RANDOM-FROM-TOP that selects at random from the top 50% subset of data with highest scores.

The RANDOM-FROM-TOP method is included only for these synthetic logistic-regression experiments, for theoretical interest as by construction, the training pool contains half of its examples from the target distribution.

The base set used for initial training contains $|U| = 4 \times 1024$ examples.

For both the scoring model and the final model training, we used 4 epochs of batch gradient descent with a linear decay learning rate scheduler. The initial learning rate was set to 0.5. We used $\epsilon = \frac{1}{10}$ for adjusting the learning rate on validation examples.

The selected subset size n was varied from 128 to 8192 in multiplicative steps of $\sqrt{2}$. All results are averaged over 10 independent runs. The final performance curves are presented in Figure 10.

E PROOF OF PROPOSITION 1

Throughout this proof, we denote by C a generic constant that can depend on c_0, C_1, M, C_η and whose value is allowed to change from line to line.

Letting $\boldsymbol{\Delta}_s(i) = \hat{\boldsymbol{\theta}}_s^{\text{bas}+i} - \hat{\boldsymbol{\theta}}_s^{\text{bas}}$, Eq. (11) yields

$$\begin{aligned}\boldsymbol{\Delta}_{k+1}(i) &= \boldsymbol{\Delta}_k(i) - \eta m \nabla^2 \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\boldsymbol{\Delta}_k(i) - \eta\nabla\ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}}; \boldsymbol{x}_i) + \mathsf{err}_k(i)\\ &= \boldsymbol{H}_k\boldsymbol{\Delta}_k(i) - \eta\nabla\ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}}; \boldsymbol{x}_i) + \mathsf{err}_k(i)\,, \end{aligned} \tag{20}$$

where $\boldsymbol{H}_k$ is defined as in Eq. (14) and

$$\mathsf{err}_k(i) := -\eta\big[\nabla\ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}+i};\boldsymbol{x}_i) - \nabla\ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}};\boldsymbol{x}_i)\big] - \eta m\int_0^1 \big[\nabla^2\widehat{R}_U(\overline{\boldsymbol{\theta}}_k(z)) - \nabla^2\widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\big]\boldsymbol{\Delta}_k(i)\mathrm{d}z\,,$$

where $\overline{\boldsymbol{\theta}}_k(z) = (1-z)\hat{\boldsymbol{\theta}}_k^{\text{bas}} + z\hat{\boldsymbol{\theta}}_k^{\text{bas}+i}$. By assumption $\boldsymbol{\theta}\mapsto\nabla\ell(\boldsymbol{\theta};\boldsymbol{x}_i)$ and $\boldsymbol{\theta}\mapsto\nabla^2\widehat{R}_U(\boldsymbol{\theta})$ are M-Lipschitz, whence

$$\begin{aligned}\|\mathsf{err}_k(i)\| &\le \eta M\|\hat{\boldsymbol{\theta}}_k^{\text{bas}+i} - \hat{\boldsymbol{\theta}}_k^{\text{bas}}\| + \eta m M\|\hat{\boldsymbol{\theta}}_k^{\text{bas}+i} - \hat{\boldsymbol{\theta}}_k^{\text{bas}}\|\|\boldsymbol{\Delta}_k(i)\|\\ &= \eta M\|\boldsymbol{\Delta}_k(i)\| + \eta m M\|\boldsymbol{\Delta}_k(i)\|^2\,. \end{aligned}\tag{21}$$

Define $\boldsymbol{\Delta}_k^{\text{lin}}(i)$ by letting $\boldsymbol{\Delta}_k^{\text{lin}}(i) = \mathbf{0}$ and, for $k\ge 0$,

$$\boldsymbol{\Delta}_{k+1}^{\text{lin}}(i) = \boldsymbol{H}_k\boldsymbol{\Delta}_k^{\text{lin}}(i) - \eta\nabla\ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}};\boldsymbol{x}_i)\,.\tag{22}$$

Comparing with Eq. (20), we obtain

$$\begin{aligned}\big(\boldsymbol{\Delta}_{k+1}(i) - \boldsymbol{\Delta}_{k+1}^{\text{lin}}(i)\big) &= \boldsymbol{H}_k\big(\boldsymbol{\Delta}_{k+1}(i) - \boldsymbol{\Delta}_{k+1}^{\text{lin}}(i)\big) + \mathsf{err}_k(\eta,m)\,,\\ \Rightarrow\ \boldsymbol{\Delta}_t(i) - \boldsymbol{\Delta}_t^{\text{lin}}(i) &= \sum_{s=0}^{t-1}\boldsymbol{M}_{t,s+1}\mathsf{err}_s(\eta,m)\,.\end{aligned}\tag{23}$$

Since $\nabla^2\widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\succeq c_0\boldsymbol{I}_d$, we have $\|\boldsymbol{H}_k\|_{\text{op}}\le(1-c_0 m\eta)$, and therefore

$$\begin{aligned}\|\boldsymbol{\Delta}_t(i) - \boldsymbol{\Delta}_t^{\text{lin}}(i)\| &\le \sum_{s=0}^{t-1}\|\boldsymbol{M}_{t,s+1}\|_{\text{op}}\|\mathsf{err}_s(\eta,m)\|\\ &\le \sum_{s=0}^{t-1}\big(1-c_0 m\eta\big)^{t-s-1}\|\mathsf{err}_s(\eta,m)\|\,.\end{aligned}\tag{24}$$

Further, from Eq. (22), and using $\|\nabla\ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}};\boldsymbol{x}_i)\|\le C_1$, we get

$$\begin{aligned}\boldsymbol{\Delta}_t^{\text{lin}}(i) &= -\eta\sum_{s=0}^{t-1}\boldsymbol{M}_{t,s+1}\nabla\ell(\hat{\boldsymbol{\theta}}_s^{\text{bas}};\boldsymbol{x}_i)\,,\\ \Rightarrow \|\boldsymbol{\Delta}_t^{\text{lin}}(i)\| &\le C_1\eta\sum_{s=0}^{t-1}\big(1-c_0 m\eta\big)^{t-s-1}\le\frac{C}{m}\,.\end{aligned}\tag{25}$$

Let $D_t(i) := \max_{s\le t}\|\boldsymbol{\Delta}_s(i)\|$, $E_t(i) := \max_{s\le t}\|\mathsf{err}_s(i)\|$. Using Eqs. (21), (24) and (25), we get

$$\begin{aligned}D_t(i) &\le \frac{C}{m} + \frac{1}{c_0 m\eta}E_{t-1}(i)\,,\\ E_t(i) &\le \eta M D_t(i) + \eta m M D_t(i)^2\,.\end{aligned}$$

Using these inequalities together, we obtain, for all $m\ge m_0$ (and eventually adjusting the constant C)

$$D_t(i)\le\frac{C}{m}\,,\quad E_t(i)\le\frac{C\eta}{m}\,,\tag{26}$$

whence, using again Eq. (24), we get

$$\|\boldsymbol{\Delta}_t(i) - \boldsymbol{\Delta}_t^{\text{lin}}(i)\|\le\frac{C}{m^2}\,.\tag{27}$$

Notice that we can rewrite

$$S_i^{\text{lin}} = -\frac{1}{L}\sum_{s=1}^{L}\langle\nabla\widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_s^{\text{bas}}),\boldsymbol{\Delta}_s^{\text{lin}}(i)\rangle\,,\tag{28}$$

whence, using the fact that $\|\nabla^2 \widehat{R}_{\text{val}}(\overline{\boldsymbol{\theta}})\|_{\text{op}} \le C$ for all $\overline{\boldsymbol{\theta}} \in [\hat{\boldsymbol{\theta}}_k^{\text{bas}}, \hat{\boldsymbol{\theta}}_k^{\text{bas}+i}]$ (this follows from the assumed bound $\|\nabla^2 \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\|_{\text{op}} \le C_1$ and the Lipschitz property of $\boldsymbol{\theta} \mapsto \nabla^2 \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}})$), we get

$$\begin{aligned}\left|S_i - S_i^{\text{lin}}\right| &\le C \max_{s\le L} \|\boldsymbol{\Delta}_s(i)\|^2 + \frac{1}{L}\sum_{s=1}^{L} \left|\langle \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_s^{\text{bas}}), \boldsymbol{\Delta}_s(i) - \boldsymbol{\Delta}_s^{\text{lin}}(i)\rangle\right| \\ &\le C \max_{s\le L} \|\boldsymbol{\Delta}_s(i)\|^2 + C \max_{s\le L} \|\boldsymbol{\Delta}_s(i) - \boldsymbol{\Delta}_s^{\text{lin}}(i)\| \\ &\le \frac{C}{m^2}\,,\end{aligned}$$

and this completes the proof.

F Proof of Theorem 1

Throughout this proof, we denote by C a generic constant that can depend on c_0, C_1, M, C_η and whose value is allowed to change from line to line.

F.1 Bound on Υ_i

The iteration for $\hat{\boldsymbol{\theta}}_k^{\text{bas}}$ and $\hat{\boldsymbol{\theta}}_k^{\text{bas,+}}$, as specified by Algorithm 2, reads

$$\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}} = \hat{\boldsymbol{\theta}}_k^{\text{bas}} - \eta m \nabla \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\,, \tag{29}$$

$$\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas,+}} = \hat{\boldsymbol{\theta}}_{0,k+1}^{\text{bas,+}} - \varepsilon\eta m_{\text{val}} \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_{0,k+1}^{\text{bas,+}})\,, \qquad \hat{\boldsymbol{\theta}}_{0,k+1}^{\text{bas,+}} = \hat{\boldsymbol{\theta}}_k^{\text{bas,+}} - \eta m \nabla \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas,+}})\,. \tag{30}$$

Letting $\boldsymbol{\Delta}_k := \hat{\boldsymbol{\theta}}_k^{\text{bas,+}} - \hat{\boldsymbol{\theta}}_k^{\text{bas}}$, and $\boldsymbol{\Delta}_{0,k} := \hat{\boldsymbol{\theta}}_{0,k}^{\text{bas,+}} - \hat{\boldsymbol{\theta}}_k^{\text{bas}}$, we obtain

$$\boldsymbol{\Delta}_{k+1} = \boldsymbol{\Delta}_{0,k+1} - \varepsilon\eta m_{\text{val}} \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}}) + \mathsf{err}_{k+1}^{(1)}\,, \tag{31}$$

$$\boldsymbol{\Delta}_{0,k+1} = \boldsymbol{H}_k \boldsymbol{\Delta}_k + \mathsf{err}_k^{(2)}\,. \tag{32}$$

where, letting $\overline{\boldsymbol{\theta}}_{0,k+1}(z) = (1-z)\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}} + z\hat{\boldsymbol{\theta}}_{0,k+1}^{\text{bas,+}}$ and $\overline{\boldsymbol{\theta}}_k(z) = (1-z)\hat{\boldsymbol{\theta}}_k^{\text{bas}} + z\hat{\boldsymbol{\theta}}_k^{\text{bas,+}}$, we have

$$\begin{aligned}\mathsf{err}_{k+1}^{(1)} &:= -\eta\varepsilon m_{\text{val}} \int_0^1 \nabla^2 \widehat{R}_{\text{val}}(\overline{\boldsymbol{\theta}}_{0,k+1}(z))\boldsymbol{\Delta}_{0,k+1}\,\mathrm{d}z\,, \\ \mathsf{err}_k^{(2)} &:= -\eta m \int_0^1 \left[\nabla^2 \widehat{R}_U(\overline{\boldsymbol{\theta}}_k(z)) - \nabla^2 \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}})\right]\boldsymbol{\Delta}_k\,\mathrm{d}z\,.\end{aligned}$$

Using the assumption that $\|\nabla^2 \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}}(z))\|_{\text{op}} \le C$ and $\boldsymbol{\theta} \mapsto \nabla^2 \widehat{R}_{\text{val}}(\boldsymbol{\theta})$ is M-Lipschitz, we get:

$$\|\mathsf{err}_{k+1}^{(1)}\| \le C\varepsilon\eta m_{\text{val}}\big\{\|\boldsymbol{\Delta}_{0,k+1}\| + \|\boldsymbol{\Delta}_{0,k+1}\|^2\big\}\,. \tag{33}$$

On the other hand, since $\boldsymbol{\theta} \mapsto \nabla^2 \widehat{R}_U(\boldsymbol{\theta})$ is also M-Lipschitz, we have

$$\|\mathsf{err}_k^{(2)}\| \le \eta m M \|\boldsymbol{\Delta}_k\|^2\,, \tag{34}$$

whence, using Eq. (32) and $\|\boldsymbol{H}_k\|_{\text{op}} \le 1$

$$\begin{aligned}&\|\boldsymbol{\Delta}_{0,k+1}\| \le \|\boldsymbol{\Delta}_k\| + \eta m M \|\boldsymbol{\Delta}_k\|^2 \\ &\Rightarrow \|\mathsf{err}_{k+1}^{(1)}\| \le C\varepsilon\eta m_{\text{val}}\big\{\|\boldsymbol{\Delta}_k\| + \|\boldsymbol{\Delta}_k\|^2 + \eta^2 m^2 \|\boldsymbol{\Delta}_k\|^4\big\}\,,\end{aligned} \tag{35}$$

where in the last line we used the assumption that $\eta m \le C_\eta$.

Substituting Eqs. (34) and (35) in Eq. (31), (32), we obtain (using again $\eta m \le C_\eta$)

$$\boldsymbol{\Delta}_{k+1} = \boldsymbol{H}_k \boldsymbol{\Delta}_k - \varepsilon\eta m_{\text{val}} \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}}) + \mathsf{err}_k\,, \tag{36}$$

$$\|\mathsf{err}_k\| \le C\eta\varepsilon m_{\text{val}}\big(\|\boldsymbol{\Delta}_k\| + \|\boldsymbol{\Delta}_k\|^4\big) + C\eta m \|\boldsymbol{\Delta}_k\|^2\,. \tag{37}$$

We define $\boldsymbol{\Delta}_k^{\text{lin}} = \mathbf{0}$ and, for $k \ge 0$,

$$\boldsymbol{\Delta}_{k+1}^{\text{lin}} = \boldsymbol{H}_k \boldsymbol{\Delta}_k^{\text{lin}} - \varepsilon\eta m_{\text{val}} \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_{k+1}^{\text{bas}})\,, \tag{38}$$

whence

$$\boldsymbol{\Delta}_t^{\text{lin}} = -\varepsilon\eta m_{\text{val}} \sum_{s=0}^{t-1} \boldsymbol{M}_{t,s+1} \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}^{\text{bas}}_{s+1})\,, \qquad \boldsymbol{\Delta}_t - \boldsymbol{\Delta}_t^{\text{lin}} = \sum_{s=0}^{t-1} \boldsymbol{M}_{t,s+1} \mathsf{err}_s\,. \tag{39}$$

Define $D_t := \max_{s\le t} \|\boldsymbol{\Delta}_s\|$, $E_t := \max_{s\le t} \|\mathsf{err}_s\|$. Using the fact that $\|\boldsymbol{M}_{t,s+1}\|_{\rm op} \le (1 - c_0 m\eta)^{t-s-1}$ and the assumption $\|\nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}^{\text{bas}}_k)\| \le C_1$, we get, from Eqs. (37), (39),

$$D_{t+1} \le \frac{C\varepsilon m_{\text{val}}}{m} + \frac{C}{m\eta} E_t\,, \tag{40}$$

$$E_t \le C\eta\varepsilon m_{\text{val}}(D_t + D_t^4) + C\eta m D_t^2\,, \tag{41}$$

Using the assumption $\varepsilon m_{\text{val}}/m \le c_*$, this is easily seen to imply

$$D_t \le C\frac{\varepsilon m_{\text{val}}}{m}\,, \qquad E_t \le C\frac{(\varepsilon m_{\text{val}})^2}{m}\eta\,. \tag{42}$$

Substituting in Eq. (39), we get

$$\|\boldsymbol{\Delta}_t - \boldsymbol{\Delta}_t^{\text{lin}}\| \le \sum_{s=0}^{t-1} (1 - c_0 m\eta)^{t-s-1} \|\mathsf{err}_s\| \le C\Big(\frac{\varepsilon m_{\text{val}}}{m}\Big)^2\,. \tag{43}$$

The linearized score of Eq. (16) can be rewritten as

$$\Upsilon_i^{\text{lin}}\varepsilon = -\frac{1}{L}\sum_{s=1}^{L} \langle \nabla \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_s; \boldsymbol{x}_i), \boldsymbol{\Delta}_s^{\text{lin}} \rangle\,. \tag{44}$$

Using the fact that $\|\nabla \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_k; \boldsymbol{x}_i)\|$, $\|\nabla \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_k; \boldsymbol{x}_i)\|_{\rm op} \le C_1$, we get

$$\begin{aligned}
\big|\Upsilon_i(\varepsilon) - \Upsilon_i^{\text{lin}}\varepsilon\big| &\le \frac{1}{L}\sum_{s=1}^{L} \Big|\ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_k; \boldsymbol{x}_i) - \ell(\hat{\boldsymbol{\theta}}^{\text{bas,+}}_k; \boldsymbol{x}_i) + \langle \nabla \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_s; \boldsymbol{x}_i), \boldsymbol{\Delta}_s^{\text{lin}} \rangle\Big| &(45)\\
&\le \frac{C}{L}\sum_{s=1}^{L} \|\boldsymbol{\Delta}_s\|^2 + \frac{C}{L}\sum_{s=1}^{L} \|\boldsymbol{\Delta}_s - \boldsymbol{\Delta}_s^{\text{lin}}\| &(46)\\
&\le C\Big(\frac{\varepsilon m_{\text{val}}}{m}\Big)^2\,, &(47)
\end{aligned}$$

F.2 Bound on ϕ_i

The iteration for $\hat{\boldsymbol{\theta}}^{\text{bas}}_k$ and $\hat{\boldsymbol{\theta}}^{\text{bas,+}}_k$, as specified by Algorithm 2, reads

$$\hat{\boldsymbol{\theta}}^{\text{bas}}_{k+1} = \hat{\boldsymbol{\theta}}^{\text{bas}}_k - \eta m \nabla \widehat{R}_U(\hat{\boldsymbol{\theta}}^{\text{bas}}_k)\,, \tag{48}$$

$$\hat{\boldsymbol{\theta}}^{\text{val}}_{k+1} = \hat{\boldsymbol{\theta}}^{\text{bas}}_{k+1} - \varepsilon\eta m_{\text{val}} \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}^{\text{bas}}_{k+1})\,. \tag{49}$$

Hence, we can rewrite

$$\phi_i^{\text{lin}}\varepsilon = -\frac{1}{L}\sum_{s=1}^{L} \langle \nabla \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_s; \boldsymbol{x}_i), \hat{\boldsymbol{\theta}}^{\text{val}}_s - \hat{\boldsymbol{\theta}}^{\text{bas}}_s \rangle\,.$$

Using the assumptions $\|\nabla^2 \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_s; \boldsymbol{x}_i)\|_{\rm op} \le C_1$, $\|\widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}^{\text{bas}}_k)\| \le C_1$, we obtain

$$\begin{aligned}
\big|\phi_i(\varepsilon) - \phi_i^{\text{lin}}\varepsilon\big| &\le \frac{1}{L}\sum_{s=1}^{L} \Big|\ell(\hat{\boldsymbol{\theta}}^{\text{val}}_s; \boldsymbol{x}_i) - \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_s; \boldsymbol{x}_i) - \langle \nabla \ell(\hat{\boldsymbol{\theta}}^{\text{bas}}_s; \boldsymbol{x}_i), \hat{\boldsymbol{\theta}}^{\text{val}}_s - \hat{\boldsymbol{\theta}}^{\text{bas}}_s \rangle\Big| &(50)\\
&\le \frac{C}{L}\sum_{s=1}^{L} \|\hat{\boldsymbol{\theta}}^{\text{val}}_s - \hat{\boldsymbol{\theta}}^{\text{bas}}_s\|^2 \le C(\varepsilon\eta m_{\text{val}})^2\,. &(51)
\end{aligned}$$

The claim thus follows by recalling that $\eta \le C_\eta/m$.

G Proof of Theorem 2

To lighten notation, we define $\boldsymbol{r}_k := \nabla \widehat{R}_{\text{val}}(\hat{\boldsymbol{\theta}}_k^{\text{bas}})$ and $\boldsymbol{v}_k(i) := \nabla \ell(\hat{\boldsymbol{\theta}}_k^{\text{bas}}; \boldsymbol{x}_i)$.

For any $L, L_1 \in \mathbb{Z}$, we have

$$
\begin{aligned}
\Upsilon_i^{\text{lin}}(L) &= \Upsilon_i^{\text{lin},0}(L) + \Upsilon_i^{\text{lin},1}(L) + \Upsilon_i^{\text{lin},2}(L) + \Upsilon_i^{\text{lin},3}(L)\,,\\
\Upsilon_i^{\text{lin},0}(L) &:= \frac{\eta m_{\text{val}}}{L} \sum_{0 \le t < s \le L_0} \langle \boldsymbol{r}_{t+1}, \boldsymbol{M}_{s,t+1}^{\mathsf{T}} \boldsymbol{v}_s(i) \rangle\,,\\
\Upsilon_i^{\text{lin},1}(L) &:= \frac{\eta m_{\text{val}}}{L} \sum_{0 \le t \le L_0, L_0 < s \le L} \langle \boldsymbol{r}_{t+1}, \boldsymbol{M}_{s,t+1}^{\mathsf{T}} \boldsymbol{v}_s(i) \rangle\,,\\
\Upsilon_i^{\text{lin},2}(L) &:= \frac{\eta m_{\text{val}}}{L} \sum_{L_0 < t < s \le L_0 : |s-t| \ge L_1} \langle \boldsymbol{r}_{t+1}, \boldsymbol{M}_{s,t+1}^{\mathsf{T}} \boldsymbol{v}_s(i) \rangle\,,\\
\Upsilon_i^{\text{lin},2}(L) &:= \frac{\eta m_{\text{val}}}{L} \sum_{L_0 < t < s \le L_0 : |s-t| \ge L_1} \langle \boldsymbol{r}_{t+1}, \boldsymbol{M}_{s,t+1}^{\mathsf{T}} \boldsymbol{v}_s(i) \rangle\,,\\
\Upsilon_i^{\text{lin},3}(L) &:= \frac{\eta m_{\text{val}}}{L} \sum_{L_0 < t < s \le L_0 : |s-t| < L_1} \langle \boldsymbol{r}_{t+1}, \boldsymbol{M}_{s,t+1}^{\mathsf{T}} \boldsymbol{v}_s(i) \rangle\,.
\end{aligned}
$$

Since by continuity we have $\lim_{k\to\infty} \nabla^2 \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}}) = \boldsymbol{Q}_\infty$, for any $\delta \in (0, 1/2)$, we can choose L_0 large enough so that $(1-\delta)\boldsymbol{Q}_\infty \preceq \nabla^2 \widehat{R}_U(\hat{\boldsymbol{\theta}}_k^{\text{bas}}) \preceq (1+\delta)\boldsymbol{Q}_\infty$ for all $k > L_0$. In particular there exists $c_0 > 0$ (independent of ε) such that $\|\boldsymbol{H}_k\|_{op} \le (1 - c_0 m\eta)$ for all $k > L_0$.

Clearly $|\Upsilon_i^{\text{lin},0}(L)| \le C(L_0)/L \to 0$ as $L \to \infty$. Further

$$
\begin{aligned}
\left|\Upsilon_i^{\text{lin},1}(L)\right| &\le \frac{C\eta m_{\text{val}}}{L} \sum_{0 \le t \le L_0, L_0 < s \le L} \|\boldsymbol{M}_{s,t+1}\|_{\text{op}}\\
&\le \frac{C\eta m_{\text{val}}}{L} \sum_{0 \le t \le L_0, L_0 < s \le L} (1 - c_0 m\eta)^{s-t-1}\\
&\le \frac{C\eta m_{\text{val}}}{L} \frac{L_0}{c_0 m\eta} \to 0\,.
\end{aligned}
$$

Finally, by increasing L_0, we can ensure that, for $k > L_0$, $\|\boldsymbol{H}_k - \boldsymbol{H}_\infty\|_{\text{op}} \le \delta$, $\|\boldsymbol{r}_k - \boldsymbol{r}_\infty\| \le \delta$, $\|\boldsymbol{v}_k(i) - \boldsymbol{v}_\infty(i)\| \le \delta$ (where $\boldsymbol{H}_\infty = \boldsymbol{I} - \eta m \boldsymbol{Q}_\infty$ and $\boldsymbol{r}_\infty$, $\boldsymbol{v}_\infty(i)$). Hence

$$
\left|\langle \boldsymbol{r}_{t+1}, \boldsymbol{M}_{s,t+1}^{\mathsf{T}} \boldsymbol{v}_s(i) \rangle - \langle \boldsymbol{r}_\infty, \boldsymbol{H}_\infty^{s-t-1} \boldsymbol{v}_\infty(i) \rangle\right| \le C|t-s+1|(1-c_0 m\eta)^{s-t-1}\delta\,.
$$

Therefore, letting

$$
\tilde{\Upsilon}_i^{\text{lin},2}(L) := \frac{\eta m_{\text{val}}}{L} \sum_{L_0 < t < s \le L} \langle \boldsymbol{r}_\infty, \boldsymbol{H}_\infty^{s-t-1} \boldsymbol{v}_\infty(i) \rangle\,, \tag{52}
$$

we have

$$
\begin{aligned}
\left|\Upsilon_i^{\text{lin},2}(L) - \tilde{\Upsilon}_i^{\text{lin},2}(L)\right| &\le \frac{\eta m_{\text{val}}}{L} \sum_{L_0 < t < s \le L} \left|\langle \boldsymbol{r}_{t+1}, \boldsymbol{M}_{s,t+1}^{\mathsf{T}} \boldsymbol{v}_s(i) \rangle - \langle \boldsymbol{r}_\infty, \boldsymbol{H}_\infty^{s-t-1} \boldsymbol{v}_\infty(i) \rangle\right|\\
&\le \frac{\eta m_{\text{val}}}{L} \sum_{L_0 < t < s \le L} C|t-s+1|(1-c_0 m\eta)^{s-t-1}\delta\\
&\le \eta m_{\text{val}} \cdot \frac{1}{(c_0 m\eta)^2}\delta\,.
\end{aligned}
$$

Finally, using again $|\langle \boldsymbol{r}_\infty, \boldsymbol{H}_\infty^{s-t-1}\boldsymbol{v}_\infty(i)\rangle| \le (1-c_0 m\eta)^{s-t-1}$, we have

$$\begin{aligned}
\lim_{L\to\infty} \tilde{\Upsilon}_i^{\text{lin},2}(L) &= \lim_{L\to\infty} \frac{\eta m_{\text{val}}}{L} \sum_{L_0<t\le L} \sum_{s=t+1}^{\infty} \langle \boldsymbol{r}_\infty, \boldsymbol{H}_\infty^{s-t-1}\boldsymbol{v}_\infty(i)\rangle \\
&= \eta m_{\text{val}} \sum_{k=0}^{\infty} \langle \boldsymbol{r}_\infty, \boldsymbol{H}_\infty^{k}\boldsymbol{v}_\infty(i)\rangle \\
&= \eta m_{\text{val}} \langle \boldsymbol{r}_\infty, (\boldsymbol{I}-\boldsymbol{H}_\infty)^{-1}\boldsymbol{v}_\infty(i)\rangle \\
&= \frac{m_{\text{val}}}{m} \langle \boldsymbol{r}_\infty, \boldsymbol{Q}_\infty^{-1}\boldsymbol{v}_\infty(i)\rangle \,.
\end{aligned}$$

This finishes the proof of the part of Eq. (18) which concerns the limit of Υ_i^{lin}. The calculation of $\lim_{L\to\infty} S_i^{\text{lin}}(L)$ is completely analogous and we omit it.

H Proof of Theorem 3

To simplify notations, we write $y_j = y(\boldsymbol{x}_j)$ for the response variables and $\boldsymbol{\psi}_j = \boldsymbol{\psi}(\boldsymbol{x}_j)$ for the feature vectors. Similarly, for the $y_j^{\text{val}} = y(\boldsymbol{z}_j^{\text{val}})$, $\boldsymbol{\psi}(\boldsymbol{z}_j^{\text{val}}) = \boldsymbol{\psi}_j^{\text{val}}$.

With these notations, we have $\boldsymbol{H}_k = \boldsymbol{H}$ independent of k and

$$\nabla\ell(\hat{\boldsymbol{\theta}};\boldsymbol{x}_i) = -(y_i - \langle \boldsymbol{\psi}_i, \boldsymbol{\theta}\rangle)\boldsymbol{\psi}_i \,, \tag{53}$$

$$\nabla\widehat{R}_{\text{val}}(\boldsymbol{\theta}) = -\frac{1}{m}\boldsymbol{\Psi}^{\mathsf{T}}\big(\boldsymbol{y}-\boldsymbol{\Psi\theta}\big)\,, \tag{54}$$

$$\boldsymbol{H} = \boldsymbol{I} - \eta\boldsymbol{\Psi}^{\mathsf{T}}\boldsymbol{\Psi}\,. \tag{55}$$

Hence,

$$\Upsilon_i^{\text{lin}} = \frac{\eta m_{\text{val}}}{L} \sum_{0\le t<s\le L} r_s(i)\langle \boldsymbol{\Psi r}_{t+1}^{\text{val}}, \boldsymbol{H}^{s-t-1}\boldsymbol{\psi}_i\rangle\,, \tag{56}$$

$$\boldsymbol{r}_t^{\text{val}} := \boldsymbol{y}^{\text{val}} - \boldsymbol{\psi}^{\text{val}}\hat{\boldsymbol{\theta}}_{t+1}^{\text{bas}}\,, \qquad r_s(i) := y_i - \langle \boldsymbol{\psi}_i, \hat{\boldsymbol{\theta}}_s^{\text{bas}}\rangle \tag{57}$$

Since $\boldsymbol{P}_{\boldsymbol{\Psi}}$ is the projector onto the null-space of $\boldsymbol{H}$, and by our choice of η, we have $\boldsymbol{H} = \boldsymbol{P}_{\boldsymbol{\Psi}} + \boldsymbol{H}_\perp$, where the row/column space of $\boldsymbol{H}_\perp$ is orthogonal to the one of $\boldsymbol{P}_{\boldsymbol{\Psi}}$ and $\|\boldsymbol{H}_\perp\|_{\text{op}} = (1-c_\psi\eta) \in [0,1)$. As a consequence $\|\boldsymbol{H}^{s-t-1} - \boldsymbol{P}_{\boldsymbol{\Psi}}\|_{\text{op}} \le (1-c_\psi\eta)^{s-t-1}$.

Define

$$\tilde{\Upsilon}_i^{\text{lin}} := \frac{\eta m_{\text{val}}}{L} \sum_{0\le t<s\le L} r_s(i)\langle \boldsymbol{\Psi r}_{t+1}^{\text{val}}, \boldsymbol{P}_{\boldsymbol{\Psi}}\boldsymbol{\psi}_i\rangle\,. \tag{58}$$

Then we have

$$\begin{aligned}
\left|\frac{1}{L}\Upsilon_i^{\text{lin}} - \frac{1}{L}\tilde{\Upsilon}_i^{\text{lin}}\right| &\le \frac{\eta m_{\text{val}}}{L^2} \sum_{0\le t<s\le L} \left| r_s(i)\langle \boldsymbol{\Psi r}_{t+1}^{\text{val}}, \boldsymbol{P}_{\boldsymbol{\Psi}}\boldsymbol{\psi}_i\rangle\right| \\
&\le \frac{\eta m_{\text{val}}}{L^2} \sum_{0\le t<s\le L} |r_s(i)|\,\|\boldsymbol{\Psi r}_{t+1}^{\text{val}}\|\|\boldsymbol{H}^{s-t-1} - \boldsymbol{P}_{\boldsymbol{\Psi}}\|_{\text{op}}\,|\boldsymbol{\psi}_i| \\
&\overset{(a)}{\le} C\frac{\eta m_{\text{val}}}{L^2} \sum_{0\le t<s\le L} (1-c_\psi\eta)^{s-t-1} \\
&\le C\frac{\eta m_{\text{val}}}{L}\frac{1}{c_\psi\eta} \overset{L\to\infty}{\longrightarrow} 0\,.
\end{aligned}$$

In (a), we used the fact that $\lim_{t\to\infty}\hat{\boldsymbol{\theta}}_t^{\text{bas}} = \hat{\boldsymbol{\theta}}$ (Bartlett et al., 2021), and therefore $|r_s(i)|$, $\|\boldsymbol{\Psi r}_{t+1}^{\text{val}}\|$ remain bounded as $s,t\to\infty$.

In view of the above, $\lim_{L\to\infty} \Upsilon_i^{\text{lin}}/L = \lim_{L\to\infty} \tilde{\Upsilon}_i^{\text{lin}}/L$. For the latter, we have

$$\begin{aligned}
\lim_{L\to\infty} \frac{1}{L}\tilde{\Upsilon}_i^{\text{lin}} &= \lim_{L\to\infty} \frac{\eta m_{\text{val}}}{L^2} \sum_{0\le t<s\le L} r_s(i)\langle \boldsymbol{\Psi} \boldsymbol{r}_{t+1}^{\text{val}}, \boldsymbol{P}_{\boldsymbol{\Psi}}\boldsymbol{\psi}_i\rangle \\
&= \lim_{L\to\infty} \frac{\eta m_{\text{val}}}{L^2} \sum_{L_0\le t<s\le L} r_s(i)\langle \boldsymbol{\Psi} \boldsymbol{r}_{t+1}^{\text{val}}, \boldsymbol{P}_{\boldsymbol{\Psi}}\boldsymbol{\psi}_i\rangle \\
&= \lim_{L\to\infty} \frac{\eta m_{\text{val}}}{L^2} \sum_{L_0\le t<s\le L} r(i)\langle \boldsymbol{\Psi} \boldsymbol{r}^{\text{val}}, \boldsymbol{P}_{\boldsymbol{\Psi}}\boldsymbol{\psi}_i\rangle + \mathsf{err}(L_0, L)\,,
\end{aligned}$$

where

$$|\mathsf{err}(L_0, L)| \le C \sup_{s\ge L_0} \|r_s(i) - r(i)| + C \sup_{t\ge L_0} \|\boldsymbol{r}_t^{\text{val}} - \boldsymbol{r}_{t+1}^{\text{val}}|\,. \tag{59}$$

Since $\lim_{t\to\infty} \hat{\boldsymbol{\theta}}_t^{\text{bas}} = \hat{\boldsymbol{\theta}}$, we have $\lim_{L_0\to\infty} \limsup_{L\to\infty} \mathsf{err}(L_0, L) = 0$. Therefore,

$$\begin{aligned}
\lim_{L\to\infty} \frac{1}{L}\tilde{\Upsilon}_i^{\text{lin}} &= \lim_{L_0\to\infty} \lim_{L\to\infty} \frac{\eta m_{\text{val}}}{L^2} \sum_{L_0\le t<s\le L} r(i)\langle \boldsymbol{\Psi} \boldsymbol{r}^{\text{val}}, \boldsymbol{P}_{\boldsymbol{\Psi}}\boldsymbol{\psi}_i\rangle \\
&= \frac{1}{2}\eta m_{\text{val}} r(i)\langle \boldsymbol{\Psi} \boldsymbol{r}^{\text{val}}, \boldsymbol{P}_{\boldsymbol{\Psi}}\boldsymbol{\psi}_i\rangle\,.
\end{aligned}$$

This proves the limit for $\Upsilon_i^{\text{lin}}(L)$ in Eq. (19).

The limit of $S_i^{\text{lin}}(L)$ is computed essentially by the same argument and we omit the derivation.

I Limitation

The core idea of "train-on-validation" impacting training examples is general, but the specific scoring function $F(.)$ and aggregation strategy might need adaptation for different problem settings.

The Score+Random selection strategy often outperformed Score-Only in our experiments, suggesting that diversity plays an important role beyond simply selecting the "most affected" examples. While this is a practical improvement, it also indicates that our current scoring mechanism might not fully capture the optimal diversity or coverage needed for effective generalization. It will be interesting to explore more sophisticated diversity-aware scoring or selection mechanisms that explicitly balance our scoring methods with representation across the data space.

Although we mitigated bias toward shorter examples through length-based binning, a more refined length-normalization or task-specific weighting might further enhance the selection process. Furthermore, it will be interesting to see if the performance of our strategies further improves compared to random selection if the learning rate is also tuned for these strategies and not just for random selection.

Finally, our theoretical analysis relies on stylized settings that are plausible for simple models but may not hold in many large-scale applications.

J Models and Datasets information

J.1 Dataset information

- Slim Orca:
  - Link
  - Citations-Longpre et al. (2023); Mukherjee et al. (2023); Lian et al. (2023)
  - Licence: mit
- Alpaca GPT-4:
  - Paper:Peng et al. (2023)
  - Repository

  - Link
  - Licence: cc-by-nc-4.0
- Alpaca GPT-3.5:
  - Paper: Taori et al. (2023)
  - Link
  - Licence: cc-by-nc-4.0
- Multinerd:
  - Paper: Tedeschi & Navigli (2022)
  - Link
  - Licence: cc-by-nc-sa-4.0
- Ai4p:
  - Link
  - Licence: link
- C4 dataset:
  - Link
  - Labeled for NER task using llms.
  - Licence: terms of use
- Syn-Big:
  - Synthetically generated by us using llms.
  - Proprietary dataset

J.2 Pretrained Model information

- Meta-Llama-3-8B AI@Meta (2024)
  - Link
  - License: llama3
- xlm-roberta-base Conneau et al. (2019)
  - Link
  - License: mit