# DATA-EFFICIENT SUPERVISED FINE-TUNING OF LANGUAGE MODELS USING OPTIMAL DESIGN

**Rohan Deb** *
University of Illinois, Urbana-Champaign
rd22@cs.illinois.edu

**Kiran Thekumparampil, Kousha Kalantari**
Amazon
{kkt,kkalant}@amazon.com

**Gaurush Hiranandani**
Typeface
gaurush@typeface.ai

**Shoham Sabach**
Amazon and Technion
ssabach@technion.ac.il

**Branislav Kveton**
Adobe Research
kveton@adobe.com

## ABSTRACT

Supervised fine-tuning (SFT) is a standard approach to adapting large language models (LLMs) to new domains. In this work, we improve the statistical efficiency of SFT by selecting an informative subset of training examples. Specifically, for a fixed budget of training examples, which determines the computational cost of fine-tuning, we determine the most informative ones. The key idea in our method is to select examples that maximize the Hessian of the log-likelihood of the LLM. We approximate it efficiently by linearizing the LLM at the last layer using multinomial logistic regression models. Our approach is computationally efficient, analyzable, and performs well empirically. We demonstrate this on several problems, and back our claims with both quantitative results and an LLM evaluation.

## 1 INTRODUCTION

*Large language models (LLMs)* (Bommasani et al., 2021) have emerged as general purpose tools that can match human performance in both zero-shot and few-shot settings (Radford et al., 2019b; Brown et al., 2020). LLMs are typically trained in three stages (Ouyang et al., 2022): pre-training on a large corpus of diverse text, supervised fine-tuning in the domain of interest (Wei et al., 2022), and alignment to human preferences (Ouyang et al., 2022; Rafailov et al., 2023). The main challenge in all stages is the sheer scale of LLMs, which increased by four orders of magnitude in just four years: from 117 million parameters in GPT-2 (2019) to 1.76 trillion parameters in GPT-4 (2023).

We focus on *supervised fine-tuning (SFT)* (Wei et al., 2022) in this work. A standard approach in SFT is to optimize a *low-rank adapter (LoRA)* (Hu et al., 2022). The key idea in LoRA is to add low-rank matrices to the matrices in the transformer layers. During fine-tuning, only the low-rank matrices are adapted. Therefore, the computational cost of LoRA is linear in the rank of the low-rank matrices, which naturally trades off the computational cost for the quality of the approximation. The simplicity of LoRA made it popular in practice and thousands of different adapters have been trained (Mangrulkar et al., 2022). We propose a complementary approach that selects a subset of most informative training examples for fine-tuning. The computational cost of fine-tuning is linear in the size of the chosen subset. Therefore, as in LoRA, the number of chosen examples naturally trades off the computational cost of fine-tuning for quality.

The idea of selecting better training examples for SFT is not new and has been explored extensively before. Coverage-based approaches select sufficiently diverse examples to form coresets Phillips (2017); Tukan et al. (2021). Quality-based sampling prioritizes weeding out low-value or unhelpful examples Wenzek et al. (2019); Muenchigoff et al. (2023). In ASK-LLM Sachdeva et al. (2024), a proxy LLM is prompted with a potential training example and asked whether the example should be used for training. We review all of these approaches in detail in Appendix A. The main difference in our work is that we choose training examples based the log-likelihood of the LLM and thus take their information value into account.

*Work done while interning at Amazon.

Without loss of generality, we view training examples in fine-tuning as sentences, each being a sequence of tokens. We want to select the most informative $n$ sentences, which determines the computational cost of fine-tuning. We do this based on the SFT objective. Specifically, note that the last layer of the LLM is a product of next token probabilities. Each probability is represented by a multinomial logistic regression model (Bishop, 2006), where the feature vector is the embedding of all previous tokens. Therefore, the problem of selecting the most informative sentences for fine-tuning can be viewed as a variant of an optimal design (Pukelsheim, 1993; Stufken & Yang, 2012) for multinomial logistic regression, where tokens in a sentence are chosen jointly based on their information value. We derive an efficient approximation to the Hessian of the LLM log-likelihood, which represents how informative a set of sentences is, and then optimize a lower bound on its log determinant to find the most informative sentences.

We make the following contributions.

**(1)** We establish a connection between the supervised fine-tuning objective of LLMs and a product of multinomial logistic regression models in Section 2.

**(2)** We propose our method in Section 3. Our main technical contribution is a computationally-efficient approximation to the log determinant of the Hessian of the log-likelihood. More specifically, all matrices in this optimization problem are $d \times d$, where $d$ is the size of transformer embeddings, as opposing to $dL \times dL$, where $L$ is the number of distinct tokens. We solve the optimization problem greedily (Nemhauser et al., 1978), using the monotonicity and submodularity of the objective. At a high level, our algorithm greedily chooses sentences with tokens that are jointly most informative. This is in a stark contrast to treating each sentence as a single data point (Das et al., 2024; Mukherjee et al., 2024; Thekumparampil et al., 2024; Liu et al., 2024; Scheid et al., 2024), which we compare to in Section 5.

**(3)** We analyze our method in Section 4. The main result of our analysis is that the prediction error of our model decreases at rate $\tilde{O}(dL/\sqrt{n})$, where $n$ is the number of chosen sentences. The dependence on $n$ is similar to other recent results in the literature (Zhu et al., 2023; Mukherjee et al., 2024; Thekumparampil et al., 2024).

**(4)** We evaluate our method empirically in Section 5. Our experiments are both synthetic and on real-world data with GPT models. We observe that our approach leads to lower prediction errors than the baselines. We also conduct a qualitative evaluation of learned GPT models by a larger LLM.

## 2 Problem Formulation

We have a dataset of $N$ sentences indexed by $i \in [N]$. A sentence $i$ consists of $M_i$ tokens indexed by $j \in [M_i]$. Let $y_{i,j}$ be the $j$-th token in sentence $i$. Each token $y_{i,j} \in [L]$ belongs to a vocabulary of size $L$. We represent sentence $i$ by the sequence of its tokens,

$$y_i \;=\; (y_{i,1},\, y_{i,2},\, \ldots,\, y_{i,M_i}),$$

and denote the entire dataset by $\mathcal{D} = \{\, y_i\}_{i \in [N]}$. To model the evolution of each sentence token-by-token, we define a vector $\mathbf{x}_{i,j} \in \mathbb{R}^d$ that captures the relevant *history* up to the $j$-th token in sentence $i$. In the simplest setting, $\mathbf{x}_{i,j}$ may be a word embedding of $y_{i,j-1}$. In a large language model, $\mathbf{x}_{i,j}$ could be the output of the pre-logit layer that encodes contextual information about tokens $y_{i,1}, y_{i,2}, \ldots, y_{i,j-1}$.

**Objective:** Our objective is to select an $n$ sized subset $\mathcal{S} \subset [N]$ of sentences from the dataset $\mathcal{D}$ and subsequently fine-tune a model using this selected set $\mathcal{S}$. For fine-tuning an LLM we use pre-logit layer embeddings of sentences to compute this subset $\mathcal{S}$.

We denote the parameter matrix by $\Theta_* \in \mathbb{R}^{d \times L}$. Its $\ell$-th column $\theta_\ell^* \in \mathbb{R}^d$ corresponds to the last-layer LLM parameters for token $\ell \in [L]$, i.e., $\Theta_* = (\theta_\ell^*)_{\ell \in [L]}$. Under a softmax model, the probability of observing token $\ell$ at position $(i, j)$ is given by

$$p\big(\ell \,\big|\, \mathbf{x}_{i,j};\, \Theta_*\big) \;=\; \frac{\exp\Big(\theta_\ell^{*\top}\, \mathbf{x}_{i,j}\Big)}{\sum\limits_{k=1}^{L} \exp\Big(\theta_k^{*\top}\, \mathbf{x}_{i,j}\Big)}, \tag{1}$$

**Algorithm 1** Greedy Optimal Design for Autoregressive Models.

**Input:** Sentences $\{\mathbf{x}_i = (\mathbf{x}_{i,j})_{j=1}^{N_i}\}_{i=1}^{N}$
Design matrix $V \leftarrow I_d$
Selected sentences $\mathcal{S} \leftarrow \emptyset$
**for** $t = 1, \ldots, n$ **do**
  $k \leftarrow \underset{i \in [N] \setminus \mathcal{S}}{\operatorname{argmax}} \log\det\left(V + \sum_{j=1}^{M_i} \mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\right)$
  $\mathcal{S} \leftarrow \mathcal{S} + \{k\}$
  $V \leftarrow V + \sum_{j=1}^{M_k} \mathbf{x}_{k,j}\mathbf{x}_{k,j}^\top$
**Output:** $\mathcal{S}$

Under such a softmax model, the goal of an autoregressive model is to learn an estimate of the unknown parameter matrix $\Theta_*$ by minimizing the negative log-likelihood

$$\mathcal{L}(\Theta) = -\frac{1}{N} \sum_{i \in [N]} \sum_{j=1}^{M_i} \log p(y_{i,j} | \mathbf{x}_{i,j}; \Theta). \tag{2}$$

Our objective is to select an $n$ sized subset $\mathcal{S} \subset [N]$ of sentences from the dataset $\mathcal{D}$ and thereafter compute the *maximum likelihood estimate (MLE)* of the parameter $\Theta_*$ on the subset $\mathcal{S}$, i.e.,

$$\min_{\Theta} \quad \mathcal{L}_{\mathcal{S}}(\Theta),$$
$$\text{where } \mathcal{L}_{\mathcal{S}}(\Theta) := -\frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j=1}^{M_i} \log p(y_{i,j} \mid \mathbf{x}_{i,j}; \Theta). \tag{3}$$

When applied to an LLM fine-tuning, we use the linearized model (with the pre-logit embeddings) only to select the subset $\mathcal{S}$ and instead of computing an MLE estimate $\hat{\Theta}$, we train all the parameters of the network.

## 3 Algorithm

The *Fisher information matrix* Fisher (1922) corresponds to the Hessian of the negative log likelihood with respect to $\Theta$ and is given by

$$\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta) = -\frac{1}{n} \sum_{i \in S} \sum_{j=1}^{M_i} \nabla^2 \log p(y_{i,j} \mid \mathbf{x}_{i,j}; \Theta)\,. \tag{4}$$

The Hessian $\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta)$ can be used to derive the covariance matrix of the MLE of $\mathcal{L}_{\mathcal{S}}(\Theta)$. Therefore, it can be used for both uncertainty quantification and information gathering. Specifically, a high-probability confidence interval on model parameters $\Theta_*$ can be typically derived using $\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta_*)$ (Abbasi-Yadkori et al., 2011; Lattimore & Szepesvari, 2019). In this work, we optimize $\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta_*)$ by maximizing all of its eigenvalues with respect to $\mathcal{S}$, which can be tractably approached as $\log\det(\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta_*))$ maximization.

This problem is hard for three reasons. First, $\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta_*)$ is a $dL \times dL$ times matrix. Therefore, for practical values of $d \approx 1000$ and $L > 100\,000$, it is computationally costly to optimize it. Second, the exact maximization is impossible because $\Theta_*$ is unknown. To address these two challenges, we derive a lower bound on $\log\det(\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta))$ that only involves $d \times d$ matrices and is $\Theta$-independent. We present the lower bound in the following lemma.

**Lemma 3.1.** *Consider the loss function described in* (3)*. Then the Hessian of the loss is given by*

$$\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j=1}^{M_i} \left( \mathbf{diag}(p(y_{i,j}|\mathbf{x}_{i,j}; \Theta)) - p(y_{i,j}|\mathbf{x}_{i,j}; \Theta) p(y_{i,j}|\mathbf{x}_{i,j}; \Theta)^\top \right) \otimes \mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top,$$

*where $\otimes$ is the tensor product. Moreover, if*

$$\mathbf{diag}(p(y_{i,j}|\mathbf{x}_{i,j};\Theta)) - p(y_{i,j}|\mathbf{x}_{i,j};\Theta)p(y_{i,j}|\mathbf{x}_{i,j};\Theta)^\top \succeq \gamma$$

*holds for some $\gamma > 0$, then*

$$\log\det(\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta)) \geq d\log\det\left(\frac{\gamma}{n}\sum_{i\in S}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\right).$$

*Proof.* The lemma is proved in Section 3.3. □

Therefore, instead of maximizing $\log\det(\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta))$, we can maximize $\log\det(\sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top)$. The last challenge is that we have a combinatorial optimization problem, choose a subset of $n$ sentences out of $N$. Since $\log\det$ is a monotone submodular function, we solve this problem greedily (Nemhauser et al., 1978).

### 3.1 Greedy Optimal Design

Our greedy algorithm is presented in Algorithm 1. We refer to the optimized Hessian as a *design matrix*, because the matrix is used to design the set of chosen sentences. The design matrix is initialized at $V = I_d$ (line 2) and the subset of selected sentences is initialized at $\mathcal{S} = \emptyset$ (line 3). In each step $t \in [n]$, the algorithm selects the sentence, from the remaining sentences $[N] \setminus \mathcal{S}$, that maximizes $\log\det$ of the design matrix of the previously chosen sentences (line 5). This sentence has the highest information gain. Intuitively, it contains the most diverse embeddings $\mathbf{x}_{i,j}^\top$ since $\log\det(V)$ can be viewed as the logarithm of the volume of an ellipsoid represented by $V$, and this is maximized when the lengths of all its axes increase equally. After the sentence is chosen, it is added to the current subset of sentences $\mathcal{S}$ (line 6) and $\sum_{j=1}^{M_i}\mathbf{x}_{k,j}\mathbf{x}_{k,j}^\top$ is added to the design matrix $V$ (line 7).

Note that Algorithm 1 selects one sentence at a time and each such iteration involves computing $\log\det$ of all remaining sentences (line 5). Such an implementation is clearly impractical. In Section 3.2, we present a computationally faster algorithm that takes advantage of the submodularity of $\log\det$ and parallelism to produce the same subset of sentences as in Algorithm 1.

We are concerned with two variants of Algorithm 1 in this work. In Section 4, we analyze it in the idealized setting where the pre-logit layer of the LLM is treated as a fixed feature vector. After Algorithm 1 collects $n$ samples, we use maximum likelihood estimation to compute the estimated model parameters

$$\hat{\Theta} = \operatorname*{argmin}_{\Theta}\mathcal{L}_{\mathcal{S}}(\Theta)\,, \tag{5}$$

where $\mathcal{L}_{\mathcal{S}}(\Theta)$ is defined in (3). We argue that $\hat{\Theta}$ approaches $\Theta_*$ as the sample size $n$ increases.

When applied to LLMs, Algorithm 1 collects $n$ sentences that are used to fine-tune an actual LLM. The embedding of the $j$-th token in sentence $i$ is the output of the pre-logit layer of the LLM, denoted by $\mathbf{x}_{i,j}$.

### 3.2 Fast Greedy Optimal Design

Now we present a more computationally-efficient variant of Algorithm 1 that exploits the submodularity of $\log\det$ and parallelism (Algorithm 2). Simply put, we implement line 5 in Algorithm 1 more efficiently, which is correspond to line 13 in Algorithm 2.

The key idea is to cache information gains, where $g_i$ is the cached information gain for sentence $i \in [N]$. The gains are initialized as $g_i \leftarrow \infty$ (line 4), updated in line 11, and we act greedily with respect to them in line 13. If the gains were always updated, note that line 13 is equivalent to line 5 in Algorithm 1, because the matrix $V$ is a constant in step $t$.

The key insight to efficient updates is that $\log\det$ is monotone and submodular. Therefore, the gains cannot increase as $V$ is updated and thus do not have to be recomputed when they are smaller than

**Algorithm 2** TokenOD: Fast Implementation of Algorithm 1

1: **Input:** Sentences $\{\mathbf{x}_i = (\mathbf{x}_{i,j})_{j=1}^{N_i}\}_{i=1}^{N}$, batch size $B$
2: Design matrix $V \leftarrow I_d$
3: Selected sentences $\mathcal{S} \leftarrow \emptyset$
4: Cached information gains $g \leftarrow \infty_N$
5: **for** $t = 1, \ldots, n$ **do**
6: $\quad g_{\max} \leftarrow 0$
7: $\quad$ **for** $b = 1, \ldots, N/B$ **do**
8: $\quad\quad \mathcal{B} \leftarrow \{(b-1)B + 1, \ldots, bB\}$
9: $\quad\quad$ **for all** $i \in \mathcal{B}$ **do**
10: $\quad\quad\quad$ **if** $g_i > g_{\max}$ **then**
11: $\quad\quad\quad\quad g_i \leftarrow \log\det\left(V + \sum_{j=1}^{M_i} \mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\right) - \log\det(V)$
12: $\quad\quad g_{\max} \leftarrow \max\{g_i\}_{i\in\mathcal{B}} + \{g_{\max}\}$
13: $\quad k \leftarrow \operatorname{argmax}_{i\in[N]\setminus\mathcal{S}} g_i$
14: $\quad \mathcal{S} \leftarrow \mathcal{S} + \{k\}$
15: $\quad V \leftarrow V + \sum_{j=1}^{M_k} \mathbf{x}_{k,j}\mathbf{x}_{k,j}^\top$
16: **Output:** $\mathcal{S}$

the highest tracked gain $g_{\max}$ at any step $t \in [n]$. We exploit this in line 10 and update $g_{\max}$ in line 12. Finally, we update $g_i$ in batches of size $B$ (line 9). This can be done in parallel and results in an additional $O(B)$ speedup. We use this implementation in our experiments and refer to it as TokenOD.

### 3.3 Proof of Lemma 3.1

In Section B Proposition B.1 we show that

$$\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}\left(\mathbf{diag}(p(y_{i,j}|\mathbf{x}_{i,j};\Theta)) - p(y_{i,j}|\mathbf{x}_{i,j};\Theta)p(y_{i,j}|\mathbf{x}_{i,j};\Theta)^\top\right)\otimes\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top$$

Now suppose for some $\gamma > 0$, $\mathbf{diag}(p(y_{i,j}|\mathbf{x}_{i,j};\Theta)) - p(y_{i,j}|\mathbf{x}_{i,j};\Theta)p(y_{i,j}|\mathbf{x}_{i,j};\Theta)^\top \succeq \gamma$. Then

$$\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta) \succeq \frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}\gamma I_L \otimes \mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top$$

where $I_L$ is the $L$ dimensional identity matrix. Therefore we have

$$\det(\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta)) \geq \det\left(I_L \otimes \frac{\gamma}{n}\sum_{i\in S}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\right)$$

Now using the fact that for $A \in \mathbb{R}^{p\times p}$ and $B \in \mathbb{R}^{q\times q}$, $\det(A\otimes B) = \det(A)^p\det(B)^q$ (See Proposition 7.1.11. in Bernstein (2009)) we have

$$\det(\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta)) \geq \det(I_L)^L\det\left(\frac{\gamma}{n}\sum_{i\in S}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\right)^d \geq \det\left(\frac{\gamma}{n}\sum_{i\in S}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\right)^d$$

Finally

$$\log\det(\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta)) \geq d\log\det\left(\frac{\gamma}{n}\sum_{i\in S}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\right),$$

completes the proof.

## 4 Error Bound

Our main Theorem 4.3 provides a $O(1/\sqrt{n})$ bound on the *maximum prediction error* of the estimated parameter $\hat{\Theta}$ constructed using the samples generated by Algorithm 1. The *maximum prediction error* is given by the following expression

$$\max_{i\in[N]} \sum_{j=1}^{M_i} \|\Theta_*^\top \mathbf{x}_{i,j} - \hat{\Theta}^\top \mathbf{x}_{i,j}\|_2.$$

Note that the *maximum prediction error* measures $\|\cdot\|_2$, i.e., it is the sum of prediction errors over the whole vocabulary, at the $j$-th token in sentence $i$, and therefore captures the error across all the $L$ words. We make the following assumption on the feature vectors and the unknown parameters.

**Assumption 4.1.** Assume that $\forall i \in [N], j \in [M_i]$, $\|\mathbf{x}_{i,j}\| \leq 1$. Further we assume that the true model parameter $\Theta_*$ satisfies $\Theta_* \in \mathcal{B}$ where

$$\mathcal{B} := \{\Theta = (\theta_\ell)_{\ell\in[L]} : \theta_\ell \in \mathbb{R}^d, \|\theta_\ell\|_2 \leq 1, \Theta\mathbb{1} = 0\}$$

Further we make a diversity assumption on our dataset. Given any arbitrary subset $\mathcal{S} \subseteq \mathcal{D}$, we define

$$\bar{\Sigma}_{\mathcal{S}} = \sigma_0 \boldsymbol{I_d} + \sum_{i\in\mathcal{S}} \sum_{j=1}^{M_i} \mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top \,, \tag{6}$$

**Assumption 4.2.** There exists a constant $\kappa \geq 1$ such that

$$\log\det(I_d + \sum_{j=1}^{M_i} \Sigma_{t-1}^{-1/2} x_{i,j} x_{i,j}^\top \Sigma_{t-1}^{-1/2}) \leq \kappa \log\det(I_d + \sum_{j=1}^{M_{I_t}} \Sigma_{t-1}^{-1/2} x_{I_t,j} x_{I_t,j}^\top \Sigma_{t-1}^{-1/2})$$

holds for any $i \in \mathcal{S}_{t-1}$ and $t \in [n]$.

**Theorem 4.3.** *Suppose Assumption 4.1 and Assumption 4.2 hold. Then for any $\delta > 0$, under the softmax model in (1), with probability $1-\delta$, the maximum prediction error of $\hat{\Theta}$ can be bounded as follows:*

$$\max_{i\in[N]} \sum_{j=1}^{M_i} \|\Theta_*^\top \mathbf{x}_{i,j} - \hat{\Theta}^\top \mathbf{x}_{i,j}\|_2 \leq CMe^2L \sqrt{\frac{\sigma_0^{-2}\log\left(1+\frac{\sigma_0^{-2}nM}{d}\right)}{\log(1+\sigma_0^{-2})}} \sqrt{\frac{d\kappa(d+\log(L/\delta))}{n}}\,.$$

*where $C > 0$ is some global constant.*

### 4.1 Proof Sketch

Suppose $\mathcal{S}$ be the subset of $n$ sentences produced by `TokenOD`. With $\hat{\Theta} = (\hat{\theta}_\ell)_{\ell\in[L]}$ and $\Theta_* = (\theta_\ell^*)_{\ell\in[L]}$ we can decompose the error as follows:

$$\begin{aligned}
\max_{i\in[N]} \sum_{j=1}^{M_i} \|\hat{\Theta}^\top \mathbf{x}_{i,j} - \Theta_*^\top \mathbf{x}_{i,j}\|_2 &\leq \max_{i\in[N]} \sum_{j=1}^{M_i} \sum_{\ell\in[L]} |(\hat{\theta}_\ell - \theta_\ell^*)^\top \mathbf{x}_{i,j}| \\
&\leq \max_{i\in[N]} \sum_{j=1}^{M_i} \sum_{\ell\in[L]} \|\hat{\theta}_\ell - \theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}} \|\mathbf{x}_{i,j}\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}} \leq \underbrace{\Big(\sum_{\ell\in[L]} \|\hat{\theta}_\ell - \theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}}\Big)}_{\text{I}} \underbrace{\max_{i\in[N]} \sum_{j=1}^{M_i} \|\mathbf{x}_{i,j}\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}}}_{\text{II}}
\end{aligned} \tag{7}$$

where $\bar{\Sigma}_{\mathcal{S}} = \sigma_0^2 \boldsymbol{I} + \sum_{i\in\mathcal{S}} \sum_{j=1}^{M_i} \mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top$. Term I corresponds to the error between the true parameter $\Theta_*$ and the MLE estimate $\hat{\Theta}$ while term II measures the maximum curvature of $\bar{\Sigma}_{\mathcal{S}}$.

Let us first consider term II. Under Assumption 4.2 we show that term II is bounded as follows.

**Lemma 4.4.** *Suppose Assumption 4.2 holds and $\mathcal{S}$ be the subset of sentences produced by Algorithm 1, $\bar{\Sigma}_{\mathcal{S}} = \sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top$ is the covariance matrix constructed using the samples in $\mathcal{S}$ and $M = \max_{i\in[N]} M_i$. Then*

$$\max_{i\in[N]}\sum_{j=1}^{M_i}\|\mathbf{x}_{i,j}\|^2_{\bar{\Sigma}_{\mathcal{S}}^{-1}} \leq \frac{\sigma_0^{-2}\log\left(1+\frac{\sigma_0^{-2}nM}{d}\right)}{\log(1+\sigma_0^{-2})}\frac{\kappa dM}{n}. \tag{8}$$

Next we need to control term I in (7). To do this we relate term I to the difference between the loss and its first order approximation as below

$$\begin{aligned}
&\mathcal{L}_{\mathcal{S}}(\hat{\Theta}) - \mathcal{L}_{\mathcal{S}}(\Theta^*) - \langle\nabla\mathcal{L}_{\mathcal{S}}(\Theta_*), \hat{\Theta}-\Theta^*\rangle \\
&\overset{(a)}{\leq} -\langle\nabla\mathcal{L}_{\mathcal{S}}(\Theta_*), \hat{\Theta}-\Theta^*\rangle \\
&= -\sum_{\ell=1}^{L}\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)^\top(\hat{\theta}_\ell-\theta_\ell^*) \\
&\overset{(b)}{\leq} \sum_{\ell=1}^{L}\left\|\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}}\|\hat{\theta}_\ell-\theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}}
\end{aligned} \tag{9}$$

where the dot product between matrices $A$ and $B$ is defined as $\langle A, B\rangle = \sum_{i,j}A_{i,j}B_{i,j}$. Inequality $(a)$ follows from $\mathcal{L}_{\mathcal{S}}(\hat{\Theta}) \leq \mathcal{L}_{\mathcal{S}}(\Theta^*)$ and $(b)$ follows from Cauchy Schwarz inequality. Next we lower bound $\mathcal{L}_{\mathcal{S}}(\hat{\Theta}) - \mathcal{L}_{\mathcal{S}}(\Theta^*) - \langle\nabla\mathcal{L}_{\mathcal{S}}(\Theta_*), \hat{\Theta}-\Theta^*\rangle$ by showing that the loss is strongly convex at $\Theta^*$ in the following lemma.

**Lemma 4.5.** *Suppose Assumption 4.1 holds and $\hat{\Theta}$ be the MLE solution as in (5) such that $\hat{\Theta}\in\mathcal{B}$. Then, there exists some $\alpha < 1$ such that*

$$\begin{aligned}
&\mathcal{L}_{\mathcal{S}}(\hat{\Theta}) - \mathcal{L}_{\mathcal{S}}(\Theta^*) - \langle\nabla\mathcal{L}_{\mathcal{S}}(\Theta_*), \hat{\Theta}-\Theta^*\rangle \\
&\geq \frac{e^{-2\alpha}}{L}\left(\sum_{\ell}\|\hat{\theta}_\ell-\theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}}\right)^2.
\end{aligned}$$

Using Lemma 4.5 and (9) we have

$$\begin{aligned}
\frac{e^{-2\alpha}}{L}\left(\sum_{\ell}\|\hat{\theta}_\ell-\theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}}\right)^2 &\leq \sum_{\ell=1}^{L}\left\|\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}}\|\hat{\theta}_\ell-\theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}} \\
&\leq \sup_{\ell\in[L]}\left\|\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}}\left(\sum_{\ell}\|\hat{\theta}_\ell-\theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}}\right)
\end{aligned}$$

and therefore

$$\left(\sum_{\ell}\|\hat{\theta}_\ell-\theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}}\right) \leq e^{2\alpha}L\sup_{\ell\in[L]}\left\|\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}} \leq e^2L\sup_{\ell\in[L]}\left\|\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}}$$

The next lemma bounds $\sup_{\ell\in[L]}\left\|\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}}$.

**Lemma 4.6.** *With probability $1-\delta$ the gradient of the loss satisfies the following bound:*

$$\sup_{\ell\in[L]}\left\|\nabla_\ell\mathcal{L}_{\mathcal{S}}(\Theta_*)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}} \leq C\sqrt{d+\log(L/\delta)} \tag{10}$$

*where $C > 0$ is some global constant.*

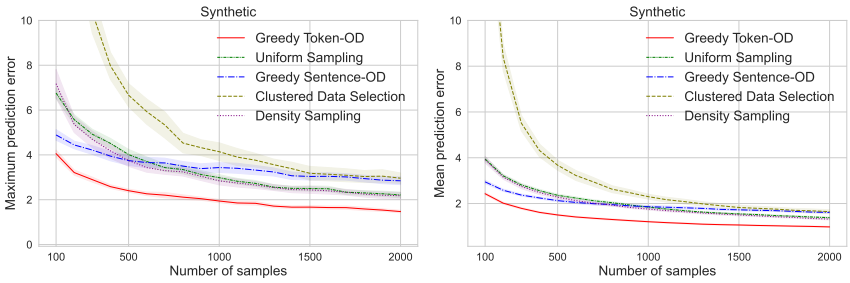


Figure 1: Comparison of maximum and mean prediction errors on synthetic token vectors. The x axis shows the number of sentences selected to train the model. The y axis shows the corresponding error averaged over 20 runs.

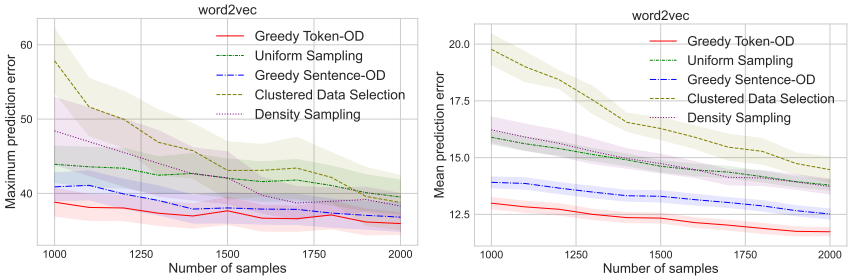


Figure 2: Comparison of maximum and mean prediction errors on word2vec token vectors. The x axis shows the number of sentences selected to train the model. The y axis shows the corresponding error averaged over 20 runs.

Combining (10), (9), (8) and (7) we have with probability $1-\delta$

$$\max_{i\in[N]} \sum_{j=1}^{M_i} \|\Theta_*^\top \mathbf{x}_{i,j} - \hat{\Theta}^\top \mathbf{x}_{i,j}\|_2$$
$$\leq CMe^2L\sqrt{\frac{\sigma_0^{-2}\log\left(1+\frac{\sigma_0^2 nM}{d}\right)}{\log(1+\sigma_0^2)}}\sqrt{\frac{d\kappa(d+\log(L/\delta))}{n}},$$

where $C>0$ is some constant, thus completing the proof.

## 5 Experiments

In this section, we empirically evaluate our algorithm and compare it to baselines. We experiment with a synthetic autoregressive prediction task in Section 5.1, with pre-trained word embeddings in Section 5.2, and with GPT-2 on text dataset in Section 5.3.

### 5.1 Synthetic Experiments

We start with a simplified setup where each token $\ell \in [L]$ is associated with a vector sampled from a standard normal distribution, $\mathbf{x}_\ell \sim \mathcal{N}(\mathbf{0}, I_d)$. The number of tokens is $L=20$ and $d=10$. The first token in each sentence is sampled uniformly at random from all tokens. Each next token is sampled the softmax model in (1), where all entries of $\Theta_*$ are sampled i.i.d. from $\mathcal{N}(0,1)$.

We compare `TokenOD` to several baselines. `Uniform` selects sentences uniformly at random. `SentenceOD` selects sentences greedily by maximizing $\log\det$ of a sentence-level Fisher information matrix. We construct sentence embeddings by summing up the token vectors in the sentences,

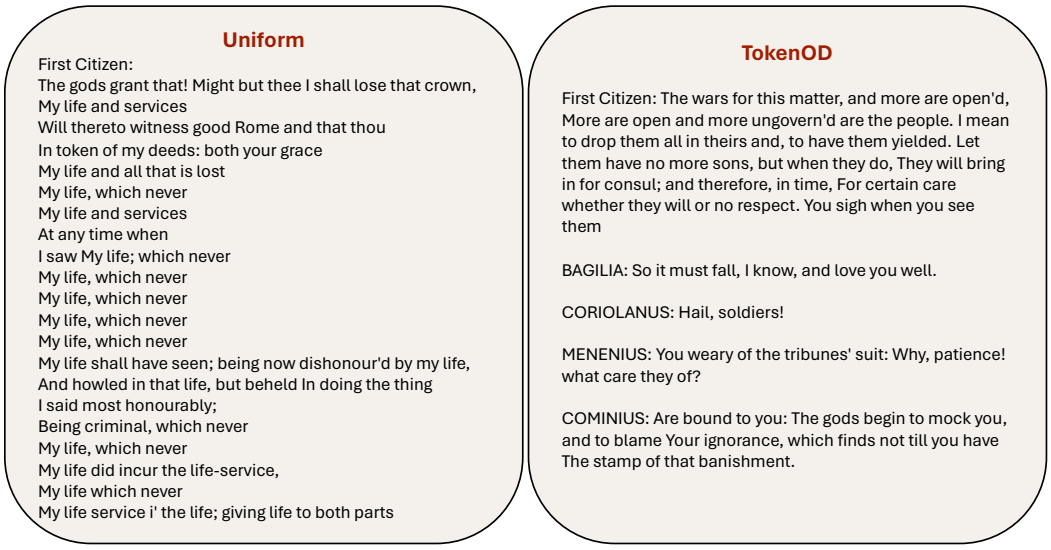


Figure 3: Text generated by fine-tuned GPT-2 models on sentences selected by `Uniform` and `TokenOD`. The latter is more coherent.

$\mathbf{x}_i = \sum_{j=1}^{M_i} \mathbf{x}_{i,j}$, where $\mathbf{x}_i$ denotes the vector for sentence $i \in [N]$. `DensitySampling` (Sachdeva et al., 2024) uses inverse propensity sampling to select sentences based on a score computed by a kernel density estimate. `ClusteredSampling` (Axiotis et al., 2024) clusters the sentence embeddings using $k$-means clustering and then samples them proportionally to their distance to the closest mean plus the mean's loss. See Appendix A for more details on the baselines.

All methods are evaluated as follows. After they choose the set of sentences $\mathcal{S}$, $\hat{\Theta}$ is estimated using multinomial logistic regression. We evaluate the methods by two metrics: *maximum prediction error*

$$\mathcal{E}_{\max}(n) = \max_{i \in [N]} \sum_{j=1}^{M_i} \|\Theta_*^\top \mathbf{x}_{i,j} - \hat{\Theta}^\top \mathbf{x}_{i,j}\|_2$$

and *mean prediction error*

$$\mathcal{E}_{mean}(n) = \frac{1}{N} \sum_{i \in [N]} \sum_{j=1}^{M_i} \|\Theta_*^\top \mathbf{x}_{i,j} - \hat{\Theta}^\top \mathbf{x}_{i,j}\|_2 \,.$$

The maximum error measures the performance on the most challenging sentence, while the mean error measures the average performance on all sentences. Note that we bound the maximum error of `TokenOD` in Theorem 4.3.

We plot the errors for our synthetic problem in Figure 1 and observe that `TokenOD` performs better than all baselines in both metrics. In most regimes, `TokenOD` is much more sample efficient than the best baseline. As an example, the lowest maximum error of the best baseline, which is attained at $n = 2\,000$, is attained by `TokenOD` at $n = 1\,000$.

### 5.2 Pre-trained Word Embeddings

The main difference in this experiment comparing to Section 5.1 is that we use pre-trained `word2vec` embeddings Mikolov et al. (2013) of dimension 300. We randomly select $L = 20$ words from the `word2vec` vocabulary and project their embeddings randomly to $d = 10$ dimensions. The vector associated with token $\ell \in [L]$ is $\mathbf{x}_\ell$. The rest is the same as in Section 5.1. We report the maximum and mean prediction errors of all methods in Figure 2. Again `TokenOD` outperforms all baselines from Section 5.1 in both metrics. As an example, the lowest mean error of the best baseline, which is attained at $n = 2\,000$, is attained by `TokenOD` at $n = 1\,500$.

Table 1: Comparison of TokenOD against various baseline data sub-sampling strategies when fine-tuning GPT-2 on the Shakespeare dataset. Entries show the fraction of times TokenOD was preferred over the corresponding baseline. All the fractions being greater than 0.5 implies that TokenOD outperforms all the baselines

| **Sampling strategy** | **Number of sentences sub-sampled for finetuning** | | | | | |
|---|---|---|---|---|---|---|
| TokenOD vs baseline | 100 | 200 | 500 | 1000 | 2000 | 5000 |
| vs Uniform | 0.80 | 0.56 | 0.60 | 0.59 | 0.64 | 0.74 |
| vs DensitySampling | 0.61 | 0.66 | 0.68 | 0.62 | 0.54 | 0.84 |
| vs AskLLM | 0.59 | 0.52 | 0.68 | 0.59 | 0.68 | 0.74 |

### 5.3 Experiments with GPT-2

**Model and Datasets:** We experiment with a tiny-Shakespeare corpus (Karpathy, 2015). Our corpus $\mathcal{D}$ is its subset of 10 000 sentences. We actively select $n \in \{50, 100, 200, 500, 1\,000, 2\,000, 5\,000\}$ sentences and then fine-tune a GPT-2 model (Radford et al., 2019a) on them. We use the Hugging Face implementation of GPT-2 (Wolf et al., 2020).

**Baselines:** We experiment with DensitySampling (Sachdeva et al., 2024), AskLLM (Sachdeva et al., 2024), and Uniform sampling baselines. See Appendix A for a detailed description of the baselines. We choose DensitySampling because it outperforms other methods on language model fine-tuning tasks (Sachdeva et al., 2024).

**LLM-based Evaluation:** Unlike in Sections 5.1 and 5.2, the ground truth model parameter is not available, and thus the maximum and mean prediction errors cannot be computed. Therefore, we judge the quality of the fine-tuned model using a state-of-the-art LLM. Specifically, we generate new text using the fine-tuned model by prompting it with 100 different phrases from the original dataset. Then we compare the generated text for two methods, say TokenOD and DensitySampling, by asking a larger GPT-4o model that serves as a judge. We use the following prompt:

```
You are a judge of Shakespeare text.
<tag1>text1</tag1>
<tag2>text2</tag2>
Respond 2 if the text inside <tag2> is more fluent Shakespeare
text than the text inside <tag1>. Respond 1 otherwise.
```

The prompt does not name the methods, and targets our perceived benefit (improved language). We use a state-of-the-art LLM GPT-4o to judge. The text generated by the compared methods is randomized: one randomly-chosen method replaces text1 and the other text2. We tested the LLM judge and it chooses the first position with probability 0.54, which is slightly higher than 0.5 for a position-unbiased judge. When comparing sentences generated by two approaches, we use the same initial phrase in each side-by-side comparison. One example of the outputs generated by the two models is in Figure 3. Clearly the model trained on uniformly selected sentences generates worse text, which is repetitive. In contrast, the text generated through TokenOD is more coherent and similar to the Shakespeare dataset. In Table 1, we report the percentage of TokenOD being preferred to the three different baselines DensitySampling, AskLLM, and Uniform for various samples sizes $n$. For all sample sizes, the text generated by the fine-tuned model on TokenOD sentences is preferred to fine-tuned models on sentences generated by the baselines.

## 6 Conclusions

In this work we developed a method to sample training examples for a fixed budget, that greedily maximizes the log determinant of the Hessian of the log likelihood. We subsequently provide a faster version of the algorithm by leveraging sub-modularity of the problem and provide bounds on the estimation error on the model trained using the collected samples. Finally through experiments on synthetic as well as real world data we evaluate our methods and show that they perform better with lower prediction errors and better quality of sentences generated by the subsequently fine-tuned models.

## REFERENCES


Ali Abbas, Karthik Tirmala, Dominic Simig, Surya Ganguli, and Ari S Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24*, pp. 2312–2320, 2011.

Kyriakos Axiotis, Vincent Cohen-Addad, Monika Henzinger, Sammy Jerome, Vahab Mirrokni, David Saulpic, David P Woodruff, and Michael Wunder. Data-efficient learning via clustering-based sensitivity sampling: Foundation models and beyond. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024.

Dennis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory*. Princeton University Press, Princeton, NJ, 2nd edition, 2009. ISBN 978-0691118028.

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

Rishi Bommasani et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL https://arxiv.org/abs/2108.07258.

Zalan Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. In *Advances in Neural Information Processing Systems*, volume 33, pp. 14879–14890, 2020.

Tom Brown et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*, 2020.

Yutian Chen, Max Welling, and Alexander Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012.

Kashyap Chitta, José M. Álvarez, Enrica Haussmann, and Emile Fardet. Training data subset search with ensemble active learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(9): 14741–14752, 2021.

Benjamin Coleman and Anshumali Shrivastava. Sub-linear race sketches for approximate kernel density estimation on streaming data. In *Proceedings of The Web Conference 2020*, WWW '20, pp. 1739–1749, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380244. URL https://doi.org/10.1145/3366423.3380244.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020.

Nirjhar Das, Souradip Chakraborty, Aldo Pacchiano, and Sayak Ray Chowdhury. Active preference optimization for sample efficient RLHF. *CoRR*, abs/2402.10500, 2024. URL https://arxiv.org/abs/2402.10500.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2881–2891, 2020.

Ronald Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London: Series A*, 222:309–368, 1922.

Bruce Hajek, Sewoong Oh, and Jiaming Xu. Minimax-optimal inference from partial rankings. 2014. URL https://arxiv.org/abs/1406.5638.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 100–108, 2014.

Zohar Karnin and Edo Liberty. Discrepancy, coreset, and sketches in machine learning. In *Conference on Learning Theory*, pp. 1975–1993. PMLR, 2019.

Andrej Karpathy. char-rnn. https://github.com/karpathy/char-rnn, 2015.

Tor Lattimore and Csaba Szepesvari. *Bandit Algorithms*. Cambridge University Press, 2019.

Alva Lee, Bruno Miranda, and Sanmi Koyejo. Beyond scale: The diversity coefficient as a data quality metric demonstrates llms are pre-trained on formally diverse data. *arXiv preprint arXiv:2306.13840*, 2023.

Kenton Lee, Daphne Ippolito, Andrew Nystrom, Cong Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, 2022.

Pangpang Liu, Chengchun Shi, and Will Wei Sun. Dual active learning for reinforcement learning from human feedback. *CoRR*, abs/2410.02504, 2024. URL https://arxiv.org/abs/2410.02504.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Robert Meding, Lars Mikkel S Buschtoff, Robert Geirhos, and Felix A Wichmann. Trivial or impossible–dichotomous data difficulty makes model differences (on imagenet and beyond). *arXiv preprint arXiv:2110.05922*, 2021.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013. URL https://api.semanticscholar.org/CorpusID:5959482.

S Mindermann, JM Brauner, MT Razzak, M Sharma, A Kirsch, W Xu, B Höltgen, AN Gomez, A Morisot, S Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022.

Michal Muenchigoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Tomasz Piktus, Nawaf Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. *arXiv preprint arXiv:2305.10623*, 2023.

Subhojyoti Mukherjee, Anusha Lalitha, Kousha Kalantari, Aniket Deshmukh, Ge Liu, Yifei Ma, and Branislav Kveton. Optimal design for human preference elicitation. In *Advances in Neural Information Processing Systems 37*, 2024.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35*, 2022.

Mikhail Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*, volume 34, pp. 2960–2971, 2021.

Jeff M Phillips. Coresets and sketches. In *Handbook of discrete and computational geometry*, pp. 1269–1288. Chapman and Hall/CRC, 2017.

Friedrich Pukelsheim. Optimal design of experiments. 1993. URL https://api.semanticscholar.org/CorpusID:122739109.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019a.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019b.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36*, 2023.

Nikhil Sachdeva, Chao-Ju Wu, and Julian McAuley. SVP-CF: Selection via proxy for collaborative filtering data. *arXiv preprint arXiv:2107.04984*, 2021.

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

Antoine Scheid, Etienne Boursier, Alain Durmus, Michael Jordan, Pierre Menard, Eric Moulines, and Michal Valko. Optimal design for reward modeling in RLHF. *CoRR*, abs/2410.17055, 2024. URL https://arxiv.org/abs/2410.17055.

Ben Sorscher, Robert Geirhos, Shreya Shekhar, Surya Ganguli, and Ari S Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 19523–19536, 2022.

John Stufken and Min Yang. Optimal designs for generalized linear models. In *Design and Analysis of Experiments*, pp. 137–164. John Wiley & Sons, 2012.

Kiran Thekumparampil, Gaurush Hiranandani, Kousha Kalantari, Shoham Sabach, and Branislav Kveton. Comparing few to rank many: Active human preference learning using randomized Frank-Wolfe. *CoRR*, abs/2412.19396, 2024. URL https://arxiv.org/abs/2412.19396.

Karthik Tirmala, Dominic Simig, Armen Aghajanyan, and Ari S Morcos. D4: Improving lm pre-training via document de-duplication and diversification. *arXiv preprint arXiv:2308.12284*, 2023.

M. Tukan, C. Baykal, D. Feldman, and D. Rus. On coresets for support vector machines. *Theoretical Computer Science*, 890:171–191, 2021.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew Dai, and Quoc Le. Finetuned language models are zero-shot learners. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Banghua Zhu, Jiantao Jiao, and Michael Jordan. Principled reinforcement learning with human feedback from pairwise or $K$-wise comparisons. *CoRR*, abs/2301.11270, 2023. URL https://arxiv.org/abs/2301.11270.

## A Related Works

**Algorithm 3** Inverse Propensity Sampling (IPS) via Kernel Density Estimation (KDE) Sachdeva et al. (2021)

1: Dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ of embeddings, sample size $k$, kernel $k$ with corresponding LSH family $\mathcal{H}$ Coleman & Shrivastava (2020), hash range $B$, rows $R$, random seed $s$.
2: Ensure a subset of $\mathcal{D}$ of size $k$, sampled with probability $p$ (see line 14).
3: Initialize KDE sketch $S \leftarrow \mathbf{0}^{R \times B}$.
4: Generate $R$ independent hash functions $h_1, \ldots, h_R$ from $\mathcal{H}$ with range $B$ and random seed $s$.
5: **for** $n \leftarrow 1$ to $N$ **do**
6:  **for** $r \leftarrow 1$ to $R$ **do**
7:  $S_{r,\, h_r(x_n)} \leftarrow S_{r,\, h_r(x_n)} + 1$
8: Initialize a list of scores $\mathcal{S} \leftarrow [\,]$.
9: **for** $n \leftarrow 1$ to $N$ **do**
10:  $score \leftarrow 0$
11:  **for** $r \leftarrow 1$ to $R$ **do**
12:  $score \leftarrow score + S[r,\, h_r(x_n)]$
13:  Append $\frac{score}{R}$ to $\mathcal{S}$.
14: **Output:** Select $k$ elements from $\mathcal{D}$ with probability $p = \frac{S}{\sum S}$ (sampled without replacement).

Coverage-oriented approaches center on ensuring that a training set reflects the entire input distribution as broadly as possible. One common strategy is *cluster sampling* Lee et al. (2023), which embeds data points in a metric space (often via learned representations) and selects mutually distant examples to form "coresets" Phillips (2017); Tukan et al. (2021). Related methods include *prototype-based sampling* for vision Sorscher et al. (2022) and *deduplication algorithms* Abbas et al. (2023); Lee et al. (2022); Tirmala et al. (2023) that remove near-duplicates or redundancies. More sophisticated procedures—such as *submodular optimization* Chen et al. (2012); Indyk et al. (2014); Borsos et al. (2020) and *discrepancy minimization* Karnin & Liberty (2019)—further refine coverage by balancing representation across diverse data regions.

Quality-based sampling, in contrast, prioritizes weeding out low-value or unhelpful examples. A prominent technique is *perplexity filtering* Wenzek et al. (2019); Muenchigoff et al. (2023), which prefers samples with higher likelihood under a pretrained model, though this can inadvertently discard valuable but rare text. Other approaches compute "uncertainty scores" via ensemble disagreement Chitta et al. (2021); Meding et al. (2021) or examine whether examples are *memorized* Feldman & Zhang (2020) or *unlearnable* Mindermann et al. (2022). The *SVP algorithm* Coleman et al. (2020); Sachdeva et al. (2021) estimates each sample's importance by its validation-loss variance, while *EL2N scores* Paul et al. (2021) track a model's difficulty in learning particular data points. These methods all fit into a "score-and-sample" framework Hastings (1970), where the final selection depends on the magnitude of each item's quality score.

For a more detailed description see Sachdeva et al. (2024). Below we describe the two algorithms proposed in Sachdeva et al. (2024) and used as benchmarks in Section 5.

**ASK-LLM:** In ASK-LLM Sachdeva et al. (2024), a proxy LLM is prompted with a potential training example and asked whether the example should be used for training. More specifically, the proxy LLM is provided the training example followed by the prompt "Does the previous paragraph contain informative signal for fine-tuning a large-language model? An informative datapoint should be well-formatted, contain some usable knowledge of the world, and strictly NOT have any harmful, racist, sexist, etc. content. OPTIONS: yes, no". It then takes the softmax probability of the token "yes" as the estimated data-quality score and sorts according to score to pick Top n data points.

**Density sampling:** Sachdeva et al. (2024) assumes access to embeddings from a pre-trained LLM. Given a dataset $D$ it uses a kernel $k(x, y)$, to estimate the density using the following score.

$$\text{score}(y) = \sum_{x \in D} k_\lambda(x, y),$$

where $\lambda$ is a smoothing parameter and controls the scale of the data points' effects. Density Sampling then uses Inverse propensity sampling (IPS) to select items proportional to their re-weighted and normalized inverse score. The algorithm as provided in Sachdeva et al. (2024) is summarized below.

**Clustering Based Sensitivity Sampling:** Axiotis et al. (2024) The method uses k-means clustering and sensitivity sampling using the embedding representation of the data with respect to which the model loss is measured and ensures that the sampled elements' average loss corresponds to the average loss of the whole dataset. The algorithm as presented in Axiotis et al. (2024) is summarized below.

**Algorithm 4** Clustering Based Sensitivity Sampling $(\mathcal{D}, k, \varepsilon, \Lambda, C)$ Axiotis et al. (2024)

1: **Input:** a dataset $\mathcal{D}$ partitioned into clusters $C = (C_1, \ldots, C_k)$ with centers $c_1, \ldots, c_k$ and a $k$-tuple of parameters $\Lambda_1, \ldots, \Lambda_k$.
2: **for** $e \in C_i$ **do**
3: Define $\hat{\ell}(e) := \ell(c_i)$ and $v(e) := \|e - c_i\|^z$.
4: Let $s := \lceil \varepsilon^{-2}(2 + 2\varepsilon/3) \rceil$. For $e \in C_i$ define $p_e := \frac{\hat{\ell}(e) + \Lambda_i v(e)}{\sum_i \Lambda_i \Phi(C_i, \{c_i\}) + \sum_{x \in \mathcal{D}} \hat{\ell}(x)}$ and $w(e) = s^{-1} p_e^{-1}$.
5: Compute a sample $S$ of $s$ points, picked independently following the distribution $p_e$.
6: **Output:** the set $S$ with weights $w$.

## B Gradient and Hessian of the Loss

**Proposition B.1.** *Consider the Loss function as defined in (3) and suppose assumption 4.1 holds. Then the gradient and Hessian of $\mathcal{L}_{\mathcal{S}}$ are respectively given by*

$$\nabla \mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in [M_i]} \text{vec}\left( \mathbf{x}_{i,j} \otimes \left( p(y_{i,j} = \ell | \mathbf{x}_{i,j}; \Theta) - \mathbb{1}(y_{i,j}) \right) \right)$$

$$\nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in [M_i]} \left( \mathbf{diag}(p(y_{i,j} | \mathbf{x}_{i,j}; \Theta)) - p(y_{i,j} | \mathbf{x}_{i,j}; \Theta) p(y_{i,j} | \mathbf{x}_{i,j}; \Theta)^\top \right) \otimes \mathbf{x}_{i,j} \mathbf{x}_{i,j}^\top$$

*Proof.* Recall that the loss function is given by

$$\begin{aligned} \mathcal{L}_{\mathcal{S}}(\Theta) &= -\frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in [M_i]} \sum_{\ell \in [L]} \log P(y_{i,j} = \ell | \mathbf{x}_{i,j}, \Theta) \delta(y_{i,j} = \ell) \\ &= -\frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in [M_i]} \sum_{\ell \in [L]} \log \left( \frac{\exp\left( (\Theta^T \mathbf{x}_{i,j})_\ell \right)}{\sum_{\ell'=1}^{L} \exp\left( (\Theta^T \mathbf{x}_{i,j})_{\ell'} \right)} \right) \delta(y_{i,j} = \ell). \end{aligned}$$

Now the loss can be re-written as

$$\mathcal{L}_{\mathcal{S}}(\Theta) = \frac{-1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in [M_i]} \left[ \theta_{y_{i,j}}^T \mathbf{x}_{i,j} - \log \sum_{\ell=1}^{L} \exp(\theta_\ell^T \mathbf{x}_n) \right]$$

Now note that

$$\frac{\partial}{\partial \theta_\ell} \theta_{y_{i,j}}^T \mathbf{x}_{i,j} = \delta(y_{i,j} = \ell) \mathbf{x}_{i,j}$$

and that,

$$\begin{aligned}\frac{\partial}{\partial\theta_\ell}\log\sum_{\ell'=1}^{L}\exp(\theta_{\ell'}^T\mathbf{x}_{i,j}) &= \frac{\sum_{\ell'=1}^{L}\exp(\theta_{\ell'}^T\mathbf{x}_{i,j})\times\delta(y_{i,j}=\ell)\mathbf{x}_{i,j}}{\sum_{k=1}^{L}\exp(\theta_k^T\mathbf{x}_{i,j})}\\ &= \sum_{\ell'=1}^{L}p(y_{i,j}=\ell'|\mathbf{x}_{i,j};\Theta)\delta(y_{i,j}=\ell)\mathbf{x}_{i,j}\\ &= p(y_{i,j}=\ell|\mathbf{x}_{i,j};\Theta)\mathbf{x}_{i,j}\end{aligned}$$

Combining both we get

$$\frac{\partial}{\partial\theta_\ell}\mathcal{L}_{\mathcal{S}}(\Theta)=\frac{-1}{n}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}\Big(\delta(y_{i,j}=\ell)-p(y_{i,j}=\ell|\mathbf{x}_{i,j};\Theta)\Big)\mathbf{x}_{i,j}$$

Therefore the gradient of the loss $\mathcal{L}_{\mathcal{S}}(\Theta)$ with respect to $\Theta$ is given by

$$\nabla\mathcal{L}_{\mathcal{S}}(\Theta)=\frac{-1}{n}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}\text{vec}\Big(\mathbf{x}_{i,j}\otimes\big(\mathbb{1}(y_{i,j})-p(y_{i,j}=\ell|\mathbf{x}_{i,j};\Theta)\big)\Big) \tag{11}$$

where $\mathbb{1}(y_{i,j})\in\mathbb{R}^L$ is a one-hot vector with the $y_{i,j}$-th entry as 1 and $\otimes$ is the Kronecker product.

Next we compute the Hessian. Note that

$$\begin{aligned}\frac{\partial^2}{\partial\theta_\ell\theta_{\ell'}}\mathcal{L}_{\mathcal{S}}(\Theta) &= \frac{-1}{n}\frac{\partial}{\partial\theta_\ell}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}\Big(\delta(y_{i,j}=\ell)-p(y_{i,j}=\ell|\mathbf{x}_{i,j};\Theta)\Big)\mathbf{x}_{i,j}\\ &= \frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}\left(\frac{\partial}{\partial\theta_{\ell'}}p(y_{i,j}=\ell|\mathbf{x}_{i,j};\Theta)\right)\mathbf{x}_{i,j}^T\\ &= \frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}p(y_{i,j}=\ell|\mathbf{x}_{i,j};\Theta)\Big(\delta(\ell=\ell')-p(y_{i,j}=\ell|\mathbf{x}_{i,j};\Theta)\Big)\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top\end{aligned}$$

and therefore, the Hessian of the loss is given by

$$\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta)=\frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}\Big(\mathbf{diag}(p(y_{i,j}|\mathbf{x}_{i,j};\Theta))-p(y_{i,j}|\mathbf{x}_{i,j};\Theta)p(y_{i,j}|\mathbf{x}_{i,j};\Theta)^\top\Big)\otimes\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top \tag{12}$$

Now note that $p(y_{i,j}|\mathbf{x}_{i,j};\Theta)\geq e^{-2\alpha}$ where $\sup_{\ell,i,j}\left|\Theta_\ell^\top\mathbf{x}_{i,j}\right|\leq\alpha$.

Therefore

$$\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta)=\frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}\Big(e^{-2\alpha}\mathbf{I}_{L\times L}-e^{-4\alpha}\mathbf{1}\mathbf{1}^\top\Big)\otimes\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top \tag{13}$$

Assume $\Big(e^{-2\alpha}\mathbf{I}_{L\times L}-e^{-4\alpha}\mathbf{1}\mathbf{1}^\top\Big)\succeq\gamma\mathbf{I}_{\mathbf{l}\times\mathbf{L}}$ for some $\gamma>0$. Then we have

$$\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta)\succeq\frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j\in[M_i]}\gamma\mathbf{I}_{\mathbf{L}\times\mathbf{L}}\otimes\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top \tag{14}$$

□

## C Proof of Error Bound

**Lemma 4.4.** *Suppose Assumption 4.2 holds and $\mathcal{S}$ be the subset of sentences produced by Algorithm 1, $\bar{\Sigma}_{\mathcal{S}}=\sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top$ is the covariance matrix constructed using the samples in $\mathcal{S}$ and $M=\max_{i\in[N]}M_i$. Then*

$$\max_{i\in[N]}\sum_{j=1}^{M_i}\|\mathbf{x}_{i,j}\|^2_{\bar{\Sigma}_{\mathcal{S}}^{-1}}\leq\frac{\sigma_0^{-2}\log\left(1+\frac{\sigma_0^{-2}nM}{d}\right)}{\log(1+\sigma_0^{-2})}\frac{\kappa dM}{n}. \tag{8}$$

*Proof.* We derive an upper bound on $\|\mathbf{x}_{i,j}\|_{\bar{\Sigma}_n^{-1}}$, where $\mathbf{x}_{i,j} \in \mathbb{R}^d$ is a feature vector and $\bar{\Sigma}_n \in \mathbb{R}^{d\times d}$ is a design matrix obtained by greedy log-determinant maximization. Let $\mathcal{D} = \{\mathbf{x}_{i,j} : i \in [N], j \in [M_i]\}$ be a dataset of $N$ data points such that $\|\mathbf{x}_{i,j}\|_2 \leq 1$. Let $I_t \in [N]$ be the index of the $t$-th chosen feature vector and $\mathcal{S}_t = \{I_\ell\}_{\ell=1}^t$ be the first $t$ chosen feature vectors. For simplicity we use $\bar{\Sigma}_{\mathcal{S}_n}$ and $\bar{\Sigma}_n$ interchangeably. Let

$$\bar{\Sigma}_t = \sigma_0^2 \boldsymbol{I} + \sum_{i\in\mathcal{S}_t}\sum_{j=1}^{M_i} \mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top$$

where $\sigma_0 > 0$ is a constant that guarantees that $\Sigma_0$ is well defined.

The $t$-th feature vector is chosen as

$$I_t = \operatorname*{argmax}_{i\in[N]\setminus\mathcal{S}_{t-1}} \log\det\left(\bar{\Sigma}_{t-1} + \sum_{j=1}^{M_t}\mathbf{x}_{t,j}\mathbf{x}_{t,j}^\top\right). \tag{15}$$

**Lemma C.1.** *For any $i \in [N]$ and $t \in [n]$,*

$$\sum_{j=1}^{M_i}\mathbf{x}_{i,j}^\top\bar{\Sigma}_t^{-1}\mathbf{x}_{i,j} \leq \sum_{j=1}^{M_i}\mathbf{x}_{i,j}^\top\bar{\Sigma}_{t-1}^{-1}\mathbf{x}_{i,j}.$$

*Proof.* Define the matrix

$$X \;=\; [\mathbf{x}_{i,1} \quad \mathbf{x}_{i,2} \quad \cdots \quad \mathbf{x}_{i,M_t}],$$

so that each $\mathbf{x}_{i,j}$ is a column of $X$. Then we can write

$$\sum_{i=1}^{M_t}\mathbf{x}_{i,j}\,\mathbf{x}_{i,j}^T \;=\; X\,X^T.$$

Hence we want to find the inverse of

$$\Sigma_{t-1} + X\,X^T.$$

Using **Sherman–Morrison–Woodbury identity**, which states that for an invertible matrix $A$ and any matrices $U, C, V$ of compatible dimensions (with $C$ also invertible), one has

$$\left(A + U\,C\,V\right)^{-1} \;=\; A^{-1} \;-\; A^{-1}\,U\left(C^{-1} + V\,A^{-1}\,U\right)^{-1}V\,A^{-1}.$$

In our case, we set

$$A = \Sigma_{t-1}, \quad U = X, \quad C = I_{M_t}, \quad V = X^T,$$

where $I_{M_t}$ is the $M_t \times M_t$ identity matrix. Then

$$A + U\,C\,V \;=\; \Sigma_{t-1} + X\,I_n\,X^T \;=\; \Sigma_{t-1} + X\,X^T.$$

By applying the identity, we get

$$(\Sigma_{t-1} + X\,X^T)^{-1} \;=\; \Sigma_{t-1}^{-1} \;-\; \Sigma_{t-1}^{-1}\,X\left(I_{M_t} + X^T\,\Sigma_{t-1}^{-1}\,X\right)^{-1}X^T\,\Sigma_{t-1}^{-1}.$$

which implies

$$\bar{\Sigma}_t^{-1} \preceq \bar{\Sigma}_{t-1}^{-1},$$

we get $v^\top\Sigma_t^{-1}v \leq v^\top\Sigma_{t-1}^{-1}v$ for any vector $v \in \mathbb{R}^d$. This concludes the proof. □

Lemma C.1 implies that

$$\sum_{j=1}^{M_i}\mathbf{x}_{i,j}^\top\bar{\Sigma}_n^{-1}\mathbf{x}_{i,j} \leq \frac{1}{n}\sum_{t=1}^{n}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}^\top\bar{\Sigma}_t^{-1}\mathbf{x}_{i,j}.$$

holds for any $i \in [N]$. This allows us to attribute the quality of the solution to individual greedy steps in (15).

If the scope of the maximization was $i \in [N]$, the inequality $\sum_{j=1}^{M_i}\mathbf{x}_{i,j}^\top\bar{\Sigma}_{t-1}^{-1}\mathbf{x}_{i,j} \leq \sum_{j=1}^{M_t}\mathbf{x}_{I_t,j}^\top\bar{\Sigma}_{t-1}^{-1}\mathbf{x}_{I_t,j}$ would hold for any $i \in [N]$. Since the scope is $i \in [N] \setminus \mathcal{S}_{t-1}$, we make Assumption 4.2.

We also use the following logarithmic transformation.

**Lemma C.2.** *For any $i \in [N]$ and $t \in [n]$,*

$$\sum_{j=1}^{M_i} \mathbf{x}_{i,j}^\top \Sigma_{t-1}^{-1} \mathbf{x}_{i,j} \leq \frac{\sigma_0^{-2} \log\left(1 + \frac{\sigma_0^{-2} nM}{d}\right)}{\log(1+\sigma_0^{-2})} \frac{\kappa d}{n} .$$

*Proof.* We start with an upper bound on $\sum_{j=1}^{M_i} \mathbf{x}_{i,j}^\top \Sigma_{t-1}^{-1} \mathbf{x}_{i,j}$. By Weyl's inequalities, we have

$$\lambda_1(\Sigma_{t-1}^{-1}) = \lambda_d^{-1}(\Sigma_{t-1}) \leq \lambda_d^{-1}(\sigma_0^2 I_d) = \sigma_0^{-2} .$$

Therefore, under the assumption that $\|\mathbf{x}_{i,j}\|_2 \leq 1$, we have $\sum_{j=1}^{M_i} \mathbf{x}_{i,j}^\top \Sigma_{t-1}^{-1} \mathbf{x}_{i,j} \leq \sigma_0^{-2} M_i$. Now note that for any $x \in [0, u]$,

$$x = \frac{x}{\log(1+x)} \log(1+x) \leq \left( \max_{x \in [0,u]} \frac{x}{\log(1+x)} \right) \log(1+x) = \frac{u}{\log(1+u)} \log(1+x) .$$

Finally, we set $x = \sum_{j=1}^{M_i} \mathbf{x}_{i,j}^\top \Sigma_{t-1}^{-1} \mathbf{x}_{i,j}$ and $u = \sigma_0^{-2} M_i$, and get our claim. □

**Assumption C.3.** There exists a constant $\kappa \geq 1$ such that

$$\log\det(I_d + \sum_{j=1}^{M_i} \Sigma_{t-1}^{-1/2} x_{i,j} x_{i,j}^\top \Sigma_{t-1}^{-1/2}) \leq \kappa \log\det(I_d + \sum_{j=1}^{M_{I_t}} \Sigma_{t-1}^{-1/2} x_{I_t,j} x_{I_t,j}^\top \Sigma_{t-1}^{-1/2})$$

holds for any $i \in \mathcal{S}_{t-1}$ and $t \in [n]$.

Now we apply Assumption C.3 and Lemma C.2, use the telescoping property of the sum, and $M = \max_{i \in [N]} M_i$ to get

$$\begin{aligned}
\sum_{t=1}^{n} \sum_{j=1}^{M_i} \mathbf{x}_{i,j}^\top \Sigma_{t-1}^{-1} \mathbf{x}_{i,j} &\leq \sum_{t=1}^{n} \sum_{j=1}^{M_i} \frac{\sigma_0^{-2}}{\log(1+\sigma_0^{-2})} \log(1 + x_{i,j}^\top \Sigma_{t-1}^{-1} x_{i,j}) \\
&\leq \frac{\sigma_0^{-2}}{\log(1+\sigma_0^{-2})} \sum_{t=1}^{n} \sum_{j=1}^{M_i} \log\det(I_d + \Sigma_{t-1}^{-1/2} x_{i,j} x_{i,j}^\top \Sigma_{t-1}^{-1/2}) \\
&\leq \frac{\sigma_0^{-2} M_i}{\log(1+\sigma_0^{-2})} \sum_{t=1}^{n} \log\det(I_d + \frac{1}{M_i} \sum_{j=1}^{M_i} \Sigma_{t-1}^{-1/2} x_{i,j} x_{i,j}^\top \Sigma_{t-1}^{-1/2}) \\
&\leq \frac{\sigma_0^{-2} M}{\log(1+\sigma_0^{-2})} \sum_{t=1}^{n} \log\det(I_d + \sum_{j=1}^{M_i} \Sigma_{t-1}^{-1/2} x_{i,j} x_{i,j}^\top \Sigma_{t-1}^{-1/2}) \\
&\leq \frac{\sigma_0^{-2} M}{\log(1+\sigma_0^{-2})} \sum_{t=1}^{n} \kappa \log\det(I_d + \sum_{j=1}^{M_{I_t}} \Sigma_{t-1}^{-1/2} x_{I_t,j} x_{I_t,j}^\top \Sigma_{t-1}^{-1/2}) \\
&= \frac{\kappa \sigma_0^{-2} M}{\log(1+\sigma_0^{-2})} \sum_{t=1}^{n} \log\det(\Sigma_{t-1} + \sum_{j=1}^{M_{I_t}} x_{I_t,j} x_{I_t,j}^\top) - \log\det(\Sigma_{t-1}) \\
&= \frac{\kappa \sigma_0^{-2} M}{\log(1+\sigma_0^{-2})} \sum_{t=1}^{n} \log\det(\Sigma_t) - \log\det(\Sigma_{t-1}) \\
&= \frac{\kappa \sigma_0^{-2} M}{\log(1+\sigma_0^{-2})} (\log\det(\Sigma_n) - \log\det(\Sigma_0)) \\
&= \frac{\kappa \sigma_0^{-2} M}{\log(1+\sigma_0^{-2})} (\log\det(\Sigma_n) - d\log(\sigma_0^2))
\end{aligned}$$

Furthermore,

$$\log\det(\Sigma_n) \le d\log\left(\frac{1}{d}\operatorname{tr}(\Sigma_n)\right) = d\log\left(1+\frac{1}{d}\sum_{t=1}^{n}\operatorname{tr}\left(\sum_{j=1}^{M_{I_t}}\mathbf{x}_{I_t,j}\mathbf{x}_{I_t,j}^\top\right)\right)$$
$$= d\log\left(\sigma_0^2 I_d + \frac{1}{d}\sum_{t=1}^{n}\sum_{j=1}^{M_{I_t}}\mathbf{x}_{I_t,j}^\top x_{I_t,j}\right) \le d\log\left(\sigma_0^2+\frac{nM}{d}\right).$$

Finally, we combine all claims and get

$$\max_{i\in[N]}\sum_{j=1}^{M_i}\mathbf{x}_{i,j}^\top\Sigma_n^{-1}\mathbf{x}_{i,j} \le \frac{\kappa}{n}\frac{\sigma_0^{-2}M}{\log(1+\sigma_0^{-2})}(d\log\det(\frac{1}{d}\operatorname{tr}(\sum_{t=1}^{n}\sum_{j=1}^{M_{I_t}}x_{i,j}x_{i,j}^\top)) - d\log(\sigma_0)) \le \frac{\sigma_0^{-2}\log\left(1+\frac{\sigma_0^{-2}nM}{d}\right)}{\log(1+\sigma_0^{-2})}\frac{\kappa d}{n}.$$

This concludes the proof.

□

**Lemma 4.5.** *Suppose Assumption 4.1 holds and $\hat{\Theta}$ be the MLE solution as in (5) such that $\hat{\Theta}\in\mathcal{B}$. Then, there exists some $\alpha<1$ such that*

$$\mathcal{L}_{\mathcal{S}}(\hat{\Theta}) - \mathcal{L}_{\mathcal{S}}(\Theta^*) - \langle\nabla\mathcal{L}_{\mathcal{S}}(\Theta_*),\hat{\Theta}-\Theta^*\rangle$$
$$\ge \frac{e^{-2\alpha}}{L}\left(\sum_{\ell}\|\hat{\theta}_\ell-\theta_\ell^*\|_{\bar{\Sigma}_{\mathcal{S}}}\right)^2.$$

*Proof.* Using Taylor's expansion

$$\mathcal{L}_{\mathcal{S}}(\Theta^*) + \langle\nabla\mathcal{L}_{\mathcal{S}}(\Theta_*),\hat{\Theta}-\Theta^*\rangle + \langle\hat{\Theta}-\Theta^*,\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta),\hat{\Theta}-\Theta^*\rangle = \mathcal{L}_{\mathcal{S}}(\hat{\Theta})$$

The Hessian is given by

$$\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}(\operatorname{diag}(\mathbf{p}_{i,j}) - \mathbf{p}_{i,j}\mathbf{p}_{i,j}^\top)\otimes(\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top)$$

where $\mathbf{p}_{i,j} = p(y_{i,j}|\mathbf{x}_{i,j};\Theta)$. Now using Claim 1 from Hajek et al. (2014) we have

$$e^{2\alpha}(\operatorname{diag}(\mathbf{p}_{i,j}) - \mathbf{p}_{i,j}\mathbf{p}_{i,j}^\top) \succeq \frac{1}{L}\mathbf{I}_L + \frac{1}{L^2}\mathbb{1}\mathbb{1}^\top$$

where $\alpha = \max_{i,j}|\theta_{*,y_{i,j}}^\top\mathbf{x}_{i,j}| \le 1$. Therefore we have

$$\nabla^2\mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}(\operatorname{diag}(\mathbf{p}_{i,j}) - \mathbf{p}_{i,j}\mathbf{p}_{i,j}^\top)\otimes(\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top)$$
$$\succeq \frac{1}{n}\sum_{i\in\mathcal{S}}\sum_{j=1}^{M_i}\left(\frac{e^{-2\alpha}}{L}\mathbf{I}_{L\times L} - \frac{e^{-2\alpha}}{L^2}\mathbb{1}\mathbb{1}^\top\right)\otimes(\mathbf{x}_{i,j}\mathbf{x}_{i,j}^\top)$$

Now consider $\langle \hat{\Theta} - \Theta^*, \nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta), \hat{\Theta} - \Theta^* \rangle$. We can express this as follows:

$$\begin{aligned}
\langle \hat{\Theta} - \Theta^*, \nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta), \hat{\Theta} - \Theta^* \rangle &= \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j=1}^{M_i} \sum_{k,k'} \left( \sqrt{\text{diag}(p_{i,j})} \Delta\Theta_{\cdot,k} \right)^\top \left( \sqrt{\text{diag}(p_{i,j})} \Delta\Theta_{\cdot,k'} \right) \left( \mathbf{x}_{i,j} \mathbf{x}_{i,j}^\top \right)_{k,k'} \\
&\quad - \langle \Delta\Theta_{\cdot,k}^\top p_{i,j}, \Delta\Theta_{\cdot,k'}^\top p_{i,j} \rangle (\mathbf{x}_{i,j} \mathbf{x}_{i,j}^\top)_{k,k'} \\
&= \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j=1}^{M_i} \Bigg( \text{Tr}\Big( \sqrt{\text{diag}(p_{i,j})} \Delta\Theta^\top (\mathbf{x}_{i,j} \mathbf{x}_{i,j}^\top) \sqrt{\text{diag}(p_{i,j})} \Delta\Theta \Big) \\
&\qquad - \text{Tr}\Big( p_{i,j}^\top \, \Delta\Theta \, \mathbf{x}_{i,j} \mathbf{x}_{i,j}^\top \, \Delta\Theta \, p_{i,j} \Big) \Bigg) \\
&= \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j=1}^{M_i} \text{Tr}\Big( \mathbf{x}_{i,j}^\top \Delta\Theta \Big( \text{diag}(p_{i,j}) - p_{i,j} p_{i,j}^\top \Big) \Delta\Theta^\top \mathbf{x}_{i,j} \Big) \\
&\geq \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j=1}^{M_i} \text{Tr}\Big( \mathbf{x}_{i,j}^\top \Delta\Theta \left( \frac{e^{-2\alpha}}{L} \mathbf{I}_{L \times L} - \frac{e^{-2\alpha}}{L^2} \mathbb{1}\mathbb{1}^\top \right) \Delta\Theta^\top \mathbf{x}_{i,j} \Big)
\end{aligned}$$

Now observe that $\Delta\Theta \mathbb{1} = 0$ follows from Assumption 4.1 and solution $\hat{\Theta}$. Therefore,

$$\begin{aligned}
\langle \hat{\Theta} - \Theta^*, \nabla^2 \mathcal{L}_{\mathcal{S}}(\Theta), \hat{\Theta} - \Theta^* \rangle &\geq \frac{e^{-2\alpha}}{nL} \sum_{i \in \mathcal{S}} \sum_{j=1}^{M_i} \text{Tr}(\Delta\Theta^\top \mathbf{x}_{i,j} \mathbf{x}_{i,j}^\top \Delta\Theta) \\
&= \frac{e^{-2\alpha}}{L} \text{Tr}(\Theta^\top \Sigma_{\mathcal{S}} \Theta) \\
&= \frac{e^{-2\alpha}}{L} \text{Tr}\Big( \Theta^\top \sqrt{\Sigma_{\mathcal{S}}} \sqrt{\Sigma_{\mathcal{S}}} \Theta \Big) \\
&= \frac{e^{-2\alpha}}{L} \|\Sigma \Delta\Theta\|_F^2 \\
&\geq \frac{e^{-2\alpha}}{L^2} \left( \sum_{\ell} \|\Delta\theta_\ell\|_{\Sigma_{\mathcal{S}}} \right)^2
\end{aligned}$$

□

**Lemma 4.6.** *With probability $1 - \delta$ the gradient of the loss satisfies the following bound:*

$$\sup_{\ell \in [L]} \left\| \nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta_*) \right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}} \leq C \sqrt{d + \log(L/\delta)} \tag{10}$$

*where $C > 0$ is some global constant.*

*Proof.* First observe that $\|\nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta)\|_{\bar{\Sigma}_{\mathcal{S}_n}^{-1}}^2 = n \|\nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta)\|_{\Sigma_{\mathcal{S}_n}^{-1}}^2$ where $\Sigma_{\mathcal{S}_n} = \frac{1}{n} \bar{\Sigma}_{\mathcal{S}_n}$. Next recall that the gradient is given by

$$\nabla \mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in [M_i]} \text{vec} \Big( \mathbf{x}_{i,j} \otimes \big( p(y_{i,j} | \mathbf{x}_{i,j}; \Theta) - \mathbb{1}(y_{i,j}) \big) \Big).$$

Therefore

$$\nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta) = \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in [M_i]} \mathbf{x}_{i,j} \big( p(y_{i,j} = \ell | \mathbf{x}_{i,j}; \Theta) - \mathbb{I}(y_{i,j} = \ell) \big).$$

Define $X \in \mathbb{R}^{nM_i \times d}$ as the matrix whose rows are $\mathbf{x}_{i,j}, i \in \mathcal{S}, j \in [M_i]$, and $V^\ell$ be the $nM_i$ dimensional vector whose entries are $p(y_{i,j} = \ell | \mathbf{x}_{i,j}; \Theta) - \mathbb{I}(y_{i,j} = \ell)$, i.e.,

$$V_{ij}^\ell = \frac{\exp(\Theta_\ell^\top \mathbf{x}_{i,j})}{\sum_{\ell'=1}^{L} \exp(\Theta_{\ell'}^\top \mathbf{x}_{i,j})} - \mathbb{I}(y_{i,j} = \ell).$$

Note that $\mathbb{E}[V^\ell] = 0$ and $\left|V^\ell_{ij}\right| \leq 2$, which implies $V$ is 4 sub-Gaussian. Therefore

$$\left\|\nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta)\right\|^2_{\Sigma_{\mathcal{S}}^{-1}} = \frac{1}{n^2}(V^\ell)^\top X \Sigma_{\mathcal{S}} X^\top V^\ell \leq \frac{1}{n}\|V^\ell\|_2^2$$

Using Bernstein's inequality, with probability $1-\delta$, for some constant $C > 0$

$$\left\|\nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta)\right\|^2_{\Sigma_{\mathcal{S}}^{-1}} \leq C\ \frac{(d + \log(1/\delta))}{n}$$

Taking a union bound over all $\ell \in [L]$ we have with probability $1-\delta$, for some constant $C > 0$

$$\sup_{\ell \in [L]} \left\|\nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta)\right\|^2_{\Sigma_{\mathcal{S}}^{-1}} \leq C\ \frac{(d + \log(L/\delta))}{n}$$

which implies we have with probability $1-\delta$, for some constant $C > 0$

$$\sup_{\ell \in [L]} \left\|\nabla_\ell \mathcal{L}_{\mathcal{S}}(\Theta)\right\|_{\bar{\Sigma}_{\mathcal{S}}^{-1}} \leq C\ \sqrt{(d + \log(L/\delta))}$$

□