SCHEDULED LEARNING WITH DECLINING DIVERSITY AND INCREMENTAL DIFFICULTY

Tianyi Zhou & Jeff Bilmes
University of Washington, Seattle
{tianyizh,bilmes}@uw.edu

ABSTRACT

We study how to adaptively select training subsets for different stages of iterative machine learning. We introduce minimax curriculum learning (MCL), which trains a model on a diverse few samples at first, and then later on a larger training set containing concentrated hard samples, thereby avoiding wasted efforts on redundant samples in early stages and on disperse outliers in later stages. At each stage, model weights and training sets are updated by solving a minimax optimization, whose objective is composed of a loss (reflecting the hardness of the training set) and a submodular regularization (measuring its diversity). MCL repeatedly solves a sequence of such optimizations with decreasing diversity and increasing training set size. Unlike the expensive alternative minimization used in previous work, we reduce MCL to minimization of a surrogate function that can be handled by submodular maximization and optimized by gradient methods. We show that MCL achieves better performance by using fewer labeled samples for both shallow and deep models.

1 INTRODUCTION

Inspired by the interaction between teacher and student in human education, recent studies (Khan et al., 2011b; Basu & Christensen, 2013; Spitkovsky et al., 2009) support that learning algorithms can be improved by updating a model on a designed sequence of training sets, i.e., a curriculum. This problem is addressed in curriculum learning (CL) (Bengio et al., 2009), where the sequence is designed by a human expert or heuristic before training begins. Instead of relying on a teacher to provide the curriculum, self-paced learning (SPL) (Kumar et al., 2010; Tang et al., 2012a; Supancic III & Ramanan, 2013; Tang et al., 2012b) learns the curriculum during the training process, by letting the student (i.e., the algorithm) determine which samples to learn from based on their hardness. Given a training set $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ of n samples and loss function $L(y_i, f(x_i, w))$, where $x_i \in \mathbb{R}^m$ represents feature vector for the ith sample, y_i is its label, and $f(x_i, w)$ is the predicted label provided by model with weight w, SPL (Kumar et al., 2010) aims at solving the following min-min optimization.

$$\min_{w \in \mathbb{R}^m} \min_{v \in [0,1]^n} \sum_{i=1}^{n} v_i L\left(y_i, f(x_i, w)\right) - \lambda \sum_{i=1}^{n} v_i. \tag{1}$$

It jointly learns model weights w and sample weights v, which are 0-1 indicators of selected samples, by alternating minimization. Fixing w, minimization w.r.t. v equals selecting samples with loss $L(y_i, f(x_i, w)) < \lambda$, where λ controls the learning amount ("learning pace" in SPL) and works as a threshold to the hardness of enrolled samples. More samples will be selected with a larger λ. Self-paced curriculum learning (Jiang et al., 2015) introduces a blending of "teacher mode" in CL and "student mode" in SPL, where the teacher can define a region of v by attaching a linear constraint $a^T v \leq c$ to Eq. (1). Evidence in previous work (Khan et al., 2011b; Tang et al., 2012b; Basu & Christensen, 2013; Bengio, 2014) shows that solving a series of Eq. (1) with increasing λ can avoid bad local minima and reduce generalization error.

Selection of training samples has also been studied in other learning settings, often with different motivations. In active learning (AL) (Settles, 2010) and experimental design (Montgomery, 2006), the learner can actively query labels of samples from an unlabeled pool during the training process, and the goal is to reduce annotation costs. Their curriculum design aims to achieve the same performance by using fewer labeled samples and ruling out redundant uninformative ones. In machine teaching

(Khan et al., 2011a; Zhu, 2015; Patil et al., 2014), the teacher designs the optimal training set so it can take the minimal efforts (e.g., the smallest number of samples) to guide the learning algorithm to find a target model. In boosting (Schapire, 1990; Freund & Schapire, 1997), the goal is to learn an ensemble of weak classifiers sequentially. It assigns large weights to samples with large loss or that are misclassified by the model in previous steps. Then the weighted error is minimized. Since the weights are not 0-1 indicators, it needs labels of all samples.

Comparing to CL and SPL, the selection criteria of the above methods do not change for different learning stages. CL and SPL use a continuation scheme (Allgower & Georg, 2003), which handles a hard task by solving a sequence of tasks from easy to hard, where the solution to each task is the "warm start" for the next harder task. In particular, they update a classifier on a sequence of training sets from easy to hard. It has been declared in (Bengio et al., 2013; Bengio, 2014) that such continuation scheme can reduce the impact of local minima when applied to neural nets. For example, after each round of alternating minimization in SPL (Eq. (1)), the threshold λ to loss $L(y_i, f(x_i, w))$ (which measures the hardness) increases and enforces selection of more samples with larger loss. The accretion of hardness leads to an increased entropy of the training set, which can also be achieved by increasing diversity. SPL with diversity (SPLD) (Jiang et al., 2014) adds to Eq. (1) a negative group sparse regularization $-\|v\|_{2,1} \triangleq -\sum_{j=1}^{b} \|v^{(j)}\|_2$ (all samples are divided into b groups and $v^{(j)}$ is the weight vector for the jth group), which favours samples from different groups.

Active learning and boosting, by contrast, always favor samples that are difficult to predict, since they are the most informative to learn. For example, uncertainty sampling (Culotta & McCallum, 2005; Scheffer et al., 2001; Dagan & Engelson, 1995; Dasgupta & Hsu, 2008) select samples that are most uncertain, while query by committee (Seung et al., 1992; Dagan & Engelson, 1995; Abe & Mamitsuka, 1998) selects the ones that multiple models most disagree on. Boosting assigns large weights to misclassified samples with large loss. Recently, diversity modeling was introduced to AL (Wei et al., 2015). It uses submodular maximization to select diverse training batches from the most uncertain samples. However, changing diversity during the learning process has not been investigated.

It is advantageous to gradually change both hardness and diversity of the training set over different learning stages. But increasing both, as in SPLD, might not help to select the most informative samples. In early stages of SPLD (Jiang et al., 2014), selected samples tend to be grouped into a few easy regions due to small diversity. However, since the prediction of the model is already precise in these regions, selecting more from them cannot yield much improvement (due to redundancy). In its later stages, difficult samples are more favored but large diversity enforces the selected ones to be dispersed over the input space. However, hard samples usually gather around the margin of local decision boundaries where the prediction has high variance. To effectively reduce the loss in these regions, we need more training samples from the same region. Moreover, diverse hard samples might be outliers, especially in later stages when the model is already "mature." Selecting them into the curriculum might make the training unstable.

Evidence for changing hardness and diversity in a curriculum can also be found in human education. For example, courses in primary and middle school usually cover a broad range of basic knowledge/skills and fundamentals of many subjects, while in college and graduate school students mainly focus on advanced topics in their majors. In addition, recent studies of bilingualism (Bialystok et al., 2012; Li et al., 2014; Mechelli et al., 2004; Kovács & Mehler, 2009) show that learning two or multiple languages in childhood is beneficial for future brain development. These indicate that large diversity can be helpful during early learning stages.

1.1 Our Approach

Motivated by these observations, we introduce a new form of curriculum learning called "minimax curriculum learning (MCL)." It increases the hardness but reduces the diversity of the curriculum during training. This is accomplished by solving a sequence of minimax optimizations of the following form with a fading weight λ and growing k.

$$\min_{w \in \mathbb{R}^m} \max_{A \subseteq V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A). \tag{2}$$

The objective is composed of the loss on a subset A of samples evaluating the hardness of A, and a normalized monotone non-decreasing submodular function $F : 2^V \to \mathbb{R}_+$ [1] of subset A measuring its diversity, where V is the ground set of all available samples. Larger loss implies that the subset A is harder to learn, while a larger $F(A)$ indicates more diversity. The weight λ controls the trade-off between diversity and hardness, while k is the "learning amount" controlling the size of A. In practice, we select k clusters rather than k samples to avoid the annotation costs and computation of loss on all samples (details are given in Section 2.3).

The submodular function $F(\cdot)$ can be chosen from a large family (e.g., including but not limited to facility location and set cover functions). All have the following diminishing returns property: given a finite ground set V, and any $A \subseteq B \subseteq V$ and a $v \notin B$,

$$F(v \cup A) - F(A) \geq F(v \cup B) - F(B). \tag{3}$$

This implies v is more important to the smaller set A than to the larger set B. The marginal gain of v conditioned on A is $f(v|A) \triangleq f(v \cup A) - f(A)$ and reflects the importance of v to A. Due to this property, submodular functions (Fujishige, 2005) have been widely used for diversity models (Batra et al., 2012; Prasad et al., 2014; Gillenwater et al., 2012; Iyer & Bilmes, 2015).

In MCL, we gradually reduce λ and augment k to increase the hardness and decrease the diversity of A, and alternatively update A and w in each iteration. At early stages, λ is large and k is small, so the maximization tends to select a few diverse samples with large $F(A)$. At later stages, λ decreases and k increases, so the maximization prefers larger but less diverse training set composed of more concentrated hard samples. This scheduling of λ and k helps to select the most informative samples for different learning stages, and saves annotation/computation on redundant ones. At early stages, rather than wasting efforts on many samples the model is already accurate on, a few diverse and representative samples are sufficiently informative to improve the model. When the model becomes "mature" after some training, the most effective way to reduce loss is to train it on more samples from where it usually fails to predict, i.e., where hard samples are located. The reduced diversity can rule out spurious hard samples, which are far away from the others and likely to be distractive outliers.

Although Eq. (2) is a hybrid optimization involving both continuous variable w and discrete variable A, it can be reduced to minimization of a piecewise function, where each piece is defined by a subset A achieving the maximum in an interval of w. It is convex when the loss is convex, so various off-the-shelf algorithms can be applied once A is known for each piece. However, the number of feasible A is $\mathcal{O}(2^n)$, and enumerating them all to find the maximum is intractable. Thanks to submodularity of the objective, fast approximate algorithms (Nemhauser et al., 1978; Minoux, 1978; Leskovec et al., 2007; Mirzasoleiman et al., 2015) exist to find a sub-optimal A. We instead minimize a surrogate of the piecewise function defined by a sub-optimal A in each interval of w. It is unknown at first but can be tracked by submodular maximization between gradient descent steps of w.

2 Minimax Curriculum Learning

The minimax problem in Eq. (2) can be explained as a two-person zero-sum game between a teacher (the maximizer) and a student (the minimizer): the teacher chooses training set A based on the student's feedback of hardness (i.e., loss achieved by current model w) and how diverse according to the teacher's expertise, while the student updates w to reduce the loss on training set A given by the teacher. Similar teacher-student interaction also exists in real life. In addition, the teacher usually introduces basic concepts at the beginning and asks easy questions from diverse topics to get sufficient feedback from the student, and then trains the student with more practices from the topics that the student finds difficult.

This minimax formulation is essentially different from the min-min formulation often used in previous CL and SPL (Kumar et al., 2010; Jiang et al., 2014; 2015). Minimizing the worst case loss is a widely used strategy in machine learning (Lanckriet et al., 2003; Farnia & Tse, 2016) to achieve better generalization performance and model robustness, especially when strong assumptions cannot be made to the data distribution. With submodular regularization, MCL can further avoid overfitting on a few outliers having large loss. Moreover, comparing to min-min formulation that alternately solves two minimizations per iteration (Alternative Convex Search (ACS) (Bazaraa et al., 1993))

[1] In this paper, we focus on $F(\cdot)$ being monotone non-decreasing with A and cardinality constraint $|A| \leq k$. But other submodular functions and constraints are also potentially interesting and worth studying.

until converge for each fixed λ, only one minimization is required to solve Eq. (2). Furthermore, SPL grows the training set up to the ground set to avoid overfitting on an easy subset. In MCL, however, the training set's size also increases but can grow much slowly (since it is diverse), and may end with a much smaller subset. In addition, introducing any diversity regularization of v to SPL usually leads to the loss of bi-convexity and requires re-design of the algorithm. Any convexity and convergence analysis, however, always hold for any convex loss and submodular regularization in MCL.

The goal of this section is to solve the minimax problem in Eq. (2), which equals to a minimization $\min_{w\in\mathbb{R}^m} g(w)$ of the following objective $g(w)$.

$$g(w) \triangleq \max_{A\subseteq V, |A|\leq k} \sum_{i\in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A) \tag{4}$$

In each region of w, an analytic form of $g(w)$ can be derived from A achieving the maximum by enumerating all possible subsets. Different regions of w are associated with different A. So $g(w)$ is piecewise convex if the loss function $L(y_i, f(x_i, w))$ is convex w.r.t. w. Unfortunately, the number of feasible subsets A is $\mathcal{O}(2^n)$ so enumerating them all is intractable. However, there exists efficient algorithms such as the greedy procedure and its variants (Nemhauser et al., 1978; Minoux, 1978) that can find a suboptimal solution to the maximization in Eq. (4) (with w fixed), which is

$$\max_{A\subseteq V, |A|\leq k} G(A) \triangleq \sum_{i\in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A). \tag{5}$$

The suboptimality is due to the submodularity of $G(A)$, since $G(A)$ is a weighted sum of a modular function $\sum_{i\in A}^{n} L\left(y_i, f(x_i, w)\right)$ and a submodular function $F(A)$. , We assume $L\left(y_i, f(x_i, w)\right) \geq 0$ w.l.o.g., so $G(A)$ is also monotone non-decreasing. For each region of w, the greedy procedure gives a sub-optimal solution $\hat{A}$ of Eq. (5) and defines a surrogate function $\hat{g}(w)$ of $g(w)$

$$\hat{g}(w) \triangleq \sum_{i\in \hat{A}} L\left(y_i, f(x_i, w)\right) + \lambda F(\hat{A}) \tag{6}$$

that satisfies $\hat{g}(w) \in [\alpha \cdot g(w), g(w)]$ (α is the approximation factor) in this region. Similar to $g(w)$, $\hat{g}(w)$ is piecewise convex if the loss function $L(y_i, f(x_i, w))$ is convex w.r.t. w. So different regions of w are associated with different $\hat{A}$. We prove in Section 2.2 that minimizing $\hat{g}(w)$ gives an approximate solution of Eq. (2).

Algorithm 1 Minimax Curriculum Learning (MCL)

input: $\pi(\cdot, \eta)$, γ, p, Δ
output: w^0
initialize: w^0, λ, k
while *not converge* **do**
  for $t \in \{0, \cdots, p\}$ **do**
    $G(A) \leftarrow \sum_{i\in A} L\left(y_i, f(x_i, w^t)\right) + \lambda F(A)$;
    $\hat{A} \leftarrow \textsc{SubmodularMax}(G, k)$;
    $\nabla \hat{g}(w^t) = \frac{\partial}{\partial w} \sum_{i\in\hat{A}} L\left(y_i, f(x_i, w^t)\right)$;
    $w^{t+1} \leftarrow w^t + \pi\left(\{w^{1:t}\}, \{\nabla \hat{g}(w^{1:t})\}, \eta\right)$;
  end for
  $w^0 \leftarrow w^p$, $\lambda \leftarrow (1-\gamma)\cdot\lambda$, $k \leftarrow k + \Delta$;
end while

With $\hat{g}(w)$ given, our algorithm is simply gradient descent for minimizing $\hat{g}(w)$, where many off-the-shelf methods can be invoked, e.g., SGD, momentum methods, Nesterov's accelerated gradient (Nesterov, 2005), Adagrad (Duchi et al., 2011), etc. The key problem is how to obtain $\hat{g}(w)$, which depends on suboptimal solutions in different regions of w. However, it is not necessary to run submodular maximization for every region of w. Since we use gradient descent methods, we only need to know $\hat{g}(w)$ for w on the optimization path. At the begining of each iteration in our algorithm, we fix w and use submodular maximization to achieve $\hat{A}$, which defines $\hat{g}(w)$ in the current region of w, then any gradient descent method is applied to $\hat{g}(w)$. Let A^* represent the optimal solution of Eq. (5), then $\hat{A}$ with approximation factor α satisfies $G(\hat{A}) \geq \alpha G(A^*)$. The greedy algorithm produces an $\hat{A}$ with $\alpha = 1 - e^{-1}$ (Nemhauser et al., 1978). Given $\hat{A}$, $\hat{g}(w)$ has gradient

$$\nabla \hat{g}(w) = \frac{\partial}{\partial w} \sum_{i\in\hat{A}} L\left(y_i, f(x_i, w)\right). \tag{7}$$

Then any gradient descent method can update w and thus minimize $\hat{g}(w)$. We can treat $\hat{A}$ as one batch if k is small, and update w by $w \leftarrow w - \eta\nabla\hat{g}(w)$ with learning rate η. For large $\hat{A}$, we use mini-batch SGD that applies the same update rule to every mini-batch of $\hat{A}$. More complex gradient descent rules $\pi(\cdot, \eta)$ can take all the historical gradients and w^t in previous steps into account. With

superscript t to index the step,

$$w^{t+1} \leftarrow w^t + \pi\left(\{w^{1:t}\}, \{\nabla \hat{g}(w^{1:t})\}, \eta\right). \tag{8}$$

Hence, each iteration of the algorithm aims to minimize $\hat{g}(w)$ and update w based on a training set $\hat{A}$ chosen by the submodular maximization. The whole algorithm (approximately) solves a sequence of Eq. (2) with decreasing λ and augmenting k, where the solution w minimizing $\hat{g}(w)$ in one iteration is the "warm-start" for the next iteration. This equals repeatedly updating the model w on a sequence of training sets $\hat{A}$ that changes from small (easy) and diverse to large set with clusters of hard samples.

Algorithm 1 gives the details of MCL. Step 5-10 aims to solve optimization in Eq. (2) with λ and k scheduled in Step 11. Step 6-7 finds a suboptimal subset $\hat{A}$ by submodular maximization, which will be discussed in Section 2.1. Step 8-9 updates w based on $\hat{A}$ by gradient descent $\pi(\cdot, \eta)$ with learning rate η. We stop the optimization after p steps to avoid overfitting. Then λ is reduced by factor $\gamma \in [0, 1]$ and k is increased by Δ. We set $p \leq 50$ due to the warm start in continuation scheme.

2.1 Submodular Maximization

We now introduce the submodular maximization algorithm used in Step 7 of Algorithm 1, whose goal is to maximize $G(A)$ in Eq. (5) and select training set $\hat{A}$ based on the hardness of samples to the current model and diversity. Though the problem in Eq. (5) is NP-hard, a near-optimal solution can be achieved by the greedy algorithm, which holds an approximation factor $\alpha = 1 - e^{-1}$ (Nemhauser et al., 1978). It starts with $A \leftarrow \emptyset$, and selects the next element with the largest marginal gain $f(v|A)$ from $V \backslash A$, i.e., $A \leftarrow A \cup \{v^*\}$ where $v^* \in \operatorname{argmax}_{v \in V \backslash A} f(v|A)$, and this repeats until $|A| = k$. It is simple to implement and usually outperforms other methods, e.g., those based on integer linear programming. However, it requires $\mathcal{O}(nk)$ function evaluations for ground set size $|V| = n$.

The lazy, or accelerated, greedy algorithm (Minoux, 1978; Leskovec et al., 2007) reduces the number of function evaluations per step by lazily updating a priority queue of marginal gains over all elements. It has the same output and guarantee as the original greedy algorithm but significantly reduces computation in practice.

Greedy and lazy greedy can guarantee a better approximation factor α better than $1 - e^{-1}$ when the objective $G(A)$ is close to modular, which is exactly the case for later stage of MCL when λ decreases to a small value. Specifically, the approximation factor is $\alpha = (1 - e^{-\kappa_G})/\kappa_G$ (Conforti & Cornuejols, 1984), which depends on the curvature $\kappa_G \in [0, 1]$ of $G(A)$ (Fujishige, 2005) below describing how modular $G(A)$ is.

$$\kappa_G \triangleq 1 - \min_{j \in V} \frac{G(j|V \backslash j)}{G(j)}. \tag{9}$$

In the extreme case of $\kappa_G = 0$, $G(A)$ is modular. As κ_G increases towards 1, $G(A)$ becomes more submodular, and the approximation factor α reduces to $1 - e^{-1}$. In MCL, κ_G decreases with the weight λ of submodular regularization $F(A)$. This increases the approximation factor α and results in the surrogate function $\hat{g}(w)$ being closer to the true objective $g(w)$.

2.2 Convergence Analysis

From the optimization perspective, Algorithm 1 uses a continuation scheme to solve a sequence of minimax problems in Eq. (2) with decreasing λ and increasing k. Each problem is solved by Step 5-10, and conveys its solution w^p as a warm start w^0 to the next problem. Instead of directly minimizing the true objective $g(w)$ in Eq. (4) that could be NP-hard to obtain, Step 5-10 minimizes a surrogate function $\hat{g}(w) \leq g(w)$ by using gradient descent rule $\pi(\cdot, \eta)$. It is interesting to study how close the solution $\hat{w}$ of applying gradient descent to $\hat{g}(w)$ until convergence approximates the real solution w^* of $\min_{w \in \mathbb{R}^m} g(w)$. In the following, we study whether running the inner loop in Step 5-10 until convergence can converge to the global solution of the minimax problem proposed in Eq. (2).

Proposition 1. *The maximum of multiple β-strongly convex functions is β-strongly convex as well.*

The proof can be found in Appendix 4.1.

Theorem 1. *(Inner-loop convergence) For minimax problem in Eq.* (2) *with given ground set of all samples V and λ, if the loss function $L\left(y_i, f(x_i, w)\right)$ is β-strongly convex and $|V| \geq k$, running*

Step 6-9 in Algorithm 1 for iterations until convergence yields a solution $\hat{w}$ satisfying

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2}{k\beta}\left(\frac{1}{\alpha} - 1\right) \cdot g(w^*), \tag{10}$$

where w^* is the optimal solution of the minimax problem in Eq.(2), $g(w^*)$ is the objective value achieved on w^*, and α is the approximation factor that submodular maximization can guarantee for $G(A)$.

The proof can be found in Appendix 4.2.

In Theorem 1, we analyze the upper bound for $\|\hat{w} - w^*\|_2^2$ based on two assumptions: 1) the loss $L\left(y_i, f(x_i, w)\right)$ being β-strongly convex w.r.t. w; and 2) $\hat{w}$ is achieved by running gradient descent in Step 6-9 until convergence. In case the loss $L\left(y_i, f(x_i, w)\right)$ is convex but not β-strongly convex, a commonly used trick to modify it to β-strongly convex is to add an ℓ_2 regularization $(\beta/2)\|w\|_2^2$. In addition, for non-convex $L\left(y_i, f(x_i, w)\right)$, it is possible to prove that with high probability, a noise perturbed SGD on $\hat{g}(w)$ can hit an ϵ-optimal local solution of $g(w)$ in polynomial time steps. We will leave this to our future works. In our empirical study (Section 3), MCL achieves compelling performance when applied to deep neural nets when loss is usually non-convex.

In Step 5-10 of Algorithm 1, we stop gradient descent after p steps rather than waiting for convergence as in Assumption 2). This is because in the continuation scheme, w^p is sufficiently good as an initialization for the next iteration, and $\hat{g}(w)$ is small enough after p steps due to the warm start from its previous iteration. We will remove this assumption in later analysis of overall convergence (i.e., Lemma 1 and Theorem 3).

In later stages of MCL when λ is small, $G(A)$ tends to be more modular, i.e., with small curvature κ_G. As discussed in Section 2.1, when κ_G is close to 0, lazy greedy can have approximation factor of $\alpha = (1 - e^{-\kappa_G})/\kappa_G$, larger than $1 - e^{-1}$ and potentially close to 1. With $g(w^*)$ upper bounded, the bound in Eq. (10) can be nearly 0. Hence, $\hat{w}$ obtained by our algorithm is sufficiently close to w^*.

In the following, we further study how the distance between the solution $\hat{w}_T$ achieved by Algorithm 1 and the global optimal solution w_T^* of the minimax problem in Eq.(2) changes with the number of iterations T in the outer-loop. We will show that different scheduling strategy for λ and k will lead to different convergence rate. This is essentially important to understand how the design of curriculum can improve the model performance.

Theorem 2. *(Outer-loop convergence) If the loss function $L\left(y_i, f(x_i, w)\right)$ is β-strongly convex, submodular function $F(\cdot)$ has curvature κ_F, the ground set V of all samples has size $|V| \geq k$, and if each inner-loop in Algorithm 1 runs Step 6-9 until convergence, then solution $\hat{w}_T$ at the end of the Tth iteration of outer-loop fulfills*

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F}{k\beta(c_1/\lambda + 1)} \leq \frac{2\kappa_F}{\beta c_1} \times \frac{\lambda}{k} \times g(w_T^*), \tag{11}$$

where w_T^* is the optimal solution of the minimax problem in Eq. (2) with λ used in the Tth iteration of outer-loop. If k starts from k_0 and linearly increases by $k \leftarrow k + \Delta$ (as in Step 11 of Algorithm 1),

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1} \times \frac{(1-\gamma)^T}{(k_0 + T\Delta)} \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^T\right], \tag{12}$$

Otherwise, if k increases exponentially, i.e., $k \leftarrow (1+\Delta) \cdot k$,

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1 k_0} \times \left(\frac{1-\gamma}{1+\Delta}\right)^T \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^T\right]. \tag{13}$$

In above results, κ_F is the curvature of submodular function $F(\cdot)$, λ_0 and k_0 are the initial values for λ and k respectively, $c_1 = \min_{j \in V, t}[L\left(y_i, f(x_i, \hat{w}_T^t)\right)/F(j)]$, $c_2 = \max_{A \subseteq V, |A| \leq k} F(A)$, and $g(w_\infty^*) = \min_{w \in \mathbb{R}^m} \max_{A \subseteq V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w)\right)$.

The proof can be found in Appendix 4.4.

Remarks: Theorem 2 gives a upper bound proportional to λ/k. Therefore, either increasing k exponentially or decreasing λ exponentially can result in a linear convergence rate. However, submodular maximization in Step 7 usually has an expensive time cost when k is large. Hence, in Algorithm 1, we choose to decrease λ exponentially but increase k linearly. Another interesting result revealed by Theorem 2 is that the constant factor is related to both the curvature κ_F of submodular term and the strongly-convex constant β of the loss term. It also depends on c_1, the minimal ratio

between per-sample loss and singular gain among all samples. We will see that the quantities κ_F/β and c_1 frequently appear in our later analysis. In addition, these two quantities are potentially interesting to analysis of more general convex-submodular hybrid optimization problems.

Although we have the convergence analysis for both the inner-loop and the outer-loop of Algorithm 1, we still need an overall convergence result for the whole algorithm, without the assumption that running the inner-loop until convergence. In particular, by indexing the itereations of inner-loop with superscript and the itereations of outer-loop with subscript, our goal is to investigate how close a solution $\hat{w}_T^p$ achieved by Algorithm 1 to a global solution $w_{\mathcal{T}}^*$ of the minimax problem Eq.(1) for a sufficiently large $\mathcal{T} \geq T$, and how the distance between them changes with p and T under different scheduling for λ and k. A larger $\mathcal{T}$ usually indicates a better performance but is also a harder target.

Lemma 1. *If the loss function $L(y_i, f(x_i, w))$ is β-strongly convex and L-smooth, submodular function $F(\cdot)$ has curvature κ_F, the ground set V of all samples has size $|V| \geq k$, and if Step 9 in Algorithm 1 uses normal gradient descent algorithm with learning rate $\eta = 1/(kL)$, running the inner-loop (Step 6-9) for p iterations yields a solution $\hat{w}^p$ satisfying*

$$\|\hat{w}^p - \hat{w}\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \|\hat{w}^0 - \hat{w}\|^2 + \frac{3\kappa_F(p+1)}{kL(c_1/\lambda + 1)} \cdot \hat{g}(\hat{w}), \tag{14}$$

where $\hat{w}$ is the solution after running the inner-loop (Step 6-9) until convergence. The above result holds for any iteration of outer-loop.

The proof can be found in Appendix 4.6.

Theorem 3. *(Overall convergence) If the loss function $L(y_i, f(x_i, w))$ is β-strongly convex and L-smooth, submodular function $F(\cdot)$ has curvature κ_F, the ground set V of all samples has size $|V| \geq k$, and if Step 9 in Algorithm 1 uses normal gradient descent algorithm with learning rate $\eta = 1/(kL)$, running the outer-loop for T iterations and each inner-loop (Step 6-9) for p iterations yields a solution $\hat{w}_T^p$, whose distance to the optimal solution $w_{\mathcal{T}}^*$ of Eq. (2) (with $\lambda = \lambda_0(1-\gamma)^{\mathcal{T}}$) for $\mathcal{T} \geq T$ fulfills*

$$\|\hat{w}_T^p - w_{\mathcal{T}}^*\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \cdot 2\|\hat{w}_T^0 - \hat{w}_T\|^2 + \frac{\lambda}{k} \cdot 2c. \tag{15}$$

For λ starting from λ_0 and exponentially decreasing by $\lambda \leftarrow (1-\gamma) \cdot \lambda$ (as in Step 11 of Algorithm 1), c is defined as

$$c \triangleq \frac{\kappa_F}{c_1}\left[\frac{3(p+1)}{L} + \frac{2}{\beta}\right] \cdot g(w_\infty^*) + \left[\frac{\kappa_F}{c_1}\left(\frac{3(p+1)}{L} + \frac{2}{\beta}\right)\lambda_0(1-\gamma)^T + \frac{2\gamma(\mathcal{T}-T)}{\beta}\right] \cdot c_2, \tag{16}$$

where c_1, c_2 and $g(w_\infty^*)$ are defined in Theorem 2.

If k starts from k_0 and linearly increases by $k \leftarrow k + \Delta$ (as in Step 11 of Algorithm 1),

$$\|\hat{w}_T^p - w_{\mathcal{T}}^*\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \cdot 2\|\hat{w}_T^0 - \hat{w}_T\|^2 + \frac{(1-\gamma)^T}{k_0 + T\Delta} \cdot 2c\lambda_0. \tag{17}$$

Otherwise, if k increases exponentially, i.e., $k \leftarrow (1+\Delta) \cdot k$,

$$\|\hat{w}_T^p - w_{\mathcal{T}}^*\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \cdot 2\|\hat{w}_T^0 - \hat{w}_T\|^2 + \left(\frac{1-\gamma}{1+\Delta}\right)^T \cdot \frac{2c\lambda_0}{k_0}. \tag{18}$$

The proof can be found in Appendix 4.7.

Remarks: In Eq. (15) of Theorem 3, the upper bound decreases exponentially with power p, so it shows the linear convergence rate of inner-loop. Moreover, a smaller c indicates a better upper bound. Interestingly, Eq. (16) shows that c is a weighted sum of $g(w_\infty^*)$ and c_2, which are respectively the optimal objective value for the minimax problem Eq. (2) without the submodular term and the optimal objective value for maximizing the submodular term only. Hence, it relates the convergence bound to the solutions of the two extreme cases. The weight for $g(w_\infty^*)$ is a constant and the weight for c_2 decreases with more outer-loop iterations T. In addition, Eq. (17) and Eq. (18) show linear convergence rate of the outer-loop. Although Theorem 3 holds for normal gradient descent, the analysis can be easily extended to accelerated gradient descent with momentum and stochastic gradient descent by substituting their update rules for $\hat{w}^{t+1}$ into the reasoning of the first term's upper bound.

2.3 ADDITIONAL EMPIRICAL IMPROVEMENTS

Step 6-7 of Algorithm 1 require computing the loss on all the available samples, and each step of submodular maximization needs to evaluate the marginal gains of all the unselected samples. This may lead to expensive computation and annotation cost (the loss computation need to know the labels) in practice. Empirically we use two tricks to achieve improvements in efficiency.

Firstly, instead of selecting individual samples into A, we select clusters. In particular, we replace the per-sample loss $L\left(y_i, f(x_i, w)\right)$ with per-cluster loss $L\left(Y^{(i)}, f(X^{(i)}, w)\right)$ that sums up losses of all samples in the cluster ($X^{(i)}$ is the ith cluster and $Y^{(i)}$ denotes the labels). To save annotation costs, we further approximate it by the loss on the sample closest to the cluster centroid, i.e.,

$$L\left(Y^{(i)}, f(X^{(i)}, w)\right) \triangleq \sum\nolimits_{j \in C^{(i)}} L\left(y_j, f(x_j, w)\right) \approx |C^{(i)}| L\left(y^{(i)}, f(x^{(i)}, w)\right), \tag{19}$$

where $C^{(i)}$ contains the indices of the samples in the cluster, and $x^{(i)}$ with label $y^{(i)}$ is the sample closest to the centroid. In practice, the loss on $x^{(i)}$ is sufficiently representative to reflect the hardness of the cluster. When computing $F(A)$ reflecting the diversity of selected clusters, we use the cluster centroid to represent each cluster. In Step 8, the gradient is computed on all the samples in the selected clusters rather than on $x^{(i)}$. By using this method, we only need to annotate and compute the loss for samples in the selected clusters and the representative samples $x^{(i)}$ of other clusters. The size of ground set in submodular maximization is also reduced to the number of clusters.

We can further reduce the ground set to save computation during submodular maximization via pruning methods, which lead to zero loss (Wei et al., 2014a) or sufficiently small loss (Zhou et al., 2017) on objective $G(A)$. In MCL, as λ decreases and $G(A)$ becomes close to modular, pruning method can rule out more elements. We provide more details and discuss other speedup methods for our submodular maximization in 4.8.

3 EXPERIMENTS

In this section, we apply different curriculum learning methods to train a logistic regression model on 20newsgroups (Lang, 1995), LeNet5 on MNIST (Lecun et al., 1998), and a convolutional neural nets (2Conv-Pool-2Conv-Pool-3DenseLayer)[2] on CIFAR10 (Krizhevsky & Hinton, 2009). Details on the datasets can be found in Table 3 of Appendix. We compare MCL and its variants to SPL (Kumar et al., 2010), SPLD (Jiang et al., 2014) and SGD with a random curriculum (i.e., with random batches). They all use mini-batch SGD $\pi(\cdot, \eta)$ with the same learning rate strategy to update the parameters w. They differ only in the curriculum, i.e., different training sequences.

Dataset	News20	MNIST	CIFAR10
SGD(random)	14.36	0.96	18.52
SPL	15.43	1.25	21.14
SPLD	16.23	1.18	20.79
MCL($\lambda, \Delta = 0$)	15.99	1.25	18.04
MCL($\Delta = 0$)	16.54	1.21	17.33
MCL+random	16.23	1.09	17.12
MCL+k	**14.12**	**0.94**	**12.87**

Table 1: Test error rate (%) of different methods (for SGD we show the lowest error of 10 random trials).

In SPL and SPLD, the training set starts from a fixed size (4000 samples for 20newsgroups, 5000 samples for MNIST and CIFAR10), and increases by a factor $\mu = 0.1$ per round of alternating minimization. The model w is updated after ρ passes of the selected training set per round. In SPLD, we further have a weight for negative group sparsity starting from ξ and increasing by a factor of 0.1 per round. We tried 5 different combinations of $\{\rho, \mu\}$ and $\{\rho, \xi\}$ for SPL and SPLD respectively. The best combination with the smallest error rate is reported. Although both SPL and SPLD can be reduced to SGD when $\lambda = 0$, we do not include this special case because SGD is already a baseline. For SGD with a random curriculum, results of 10 independent trials are reported.

Dataset	News20	MNIST	CIFAR10
Total time	2649.19s	3418.97s	3677.73s
SUBMODULARMAX	62.44s	35.33s	127.36s

Table 2: Total time (seconds) of MCL+k and the time spent only on SUBMODULARMAX.

In our MCL experiments, a feature based submodular function (Wei et al., 2014b) is used for regularization, i.e., $F(A) = \sum_{u \in \mathcal{U}} \omega_u \sqrt{c_u(A)}$, where $\mathcal{U}$ is a set of features. For a subset A of clusters, $c_u(A) = \sum_{i \in A} c_u(i)$, where $c_u(i)$ is the nonnegative score of centroid for cluster i. We use

[2]The "v3" network from `https://github.com/jseppanen/cifar_lasagne`.

TF-IDF features for 20newsgroup, and the input feature to the output layer (given by ReLU) for MNIST and CIFAR10. The submodularity of $F(A)$ holds because these vectors are nonnegative.

We consider four variants of MCL: 1) MCL with $\lambda = 0$ and $\Delta = 0$, having neither submodular regularization that promotes diversity nor scheduling of k that increases hardness; 2) MCL with $\Delta = 0$, without submodular regularization but with scheduling of k; 3) MCL+random, which inserts one round that randomly samples r clusters as training set $\hat{A}$ after every q rounds of the outer loop in Algorithm 1; 4) MCL+k, which has scheduling of λ and k, but does not use a random training set. We tried 5 different combinations of $\{q, r\}$ for MCL+random and 5 different Δ for MCL+k, and report the one with the smallest test error. Other parameters such as initial values for λ, k, γ, p and the total number of clusters are the same for different variants (exact values are given in Table 4 of Appendix.

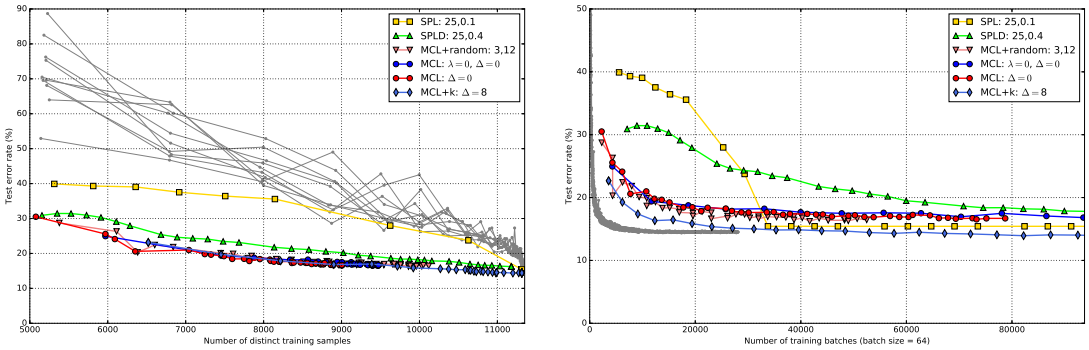

Figure 1: Test error rate (%) vs. number of distinct training sample (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

In MCL, SUBMODULARMAX is the only extra computation comparing to normal SGD. To show that its time cost is ignorable (but still brings a clear advantage as shown later), we report the total time cost of MCL+k and the time spent on SUBMODULARMAX in Table 2.

We summarize the main results in Figure 1-4. More results are given in the end of Appendix. In all figures, grey curves correspond to the 10 trials of SGD with a random curriculum. The legend gives the parameters used in different methods of the following formats: 1) SPL: ρ, μ; 2) SPLD: ρ, ξ; 3) MCL+random: q, r.

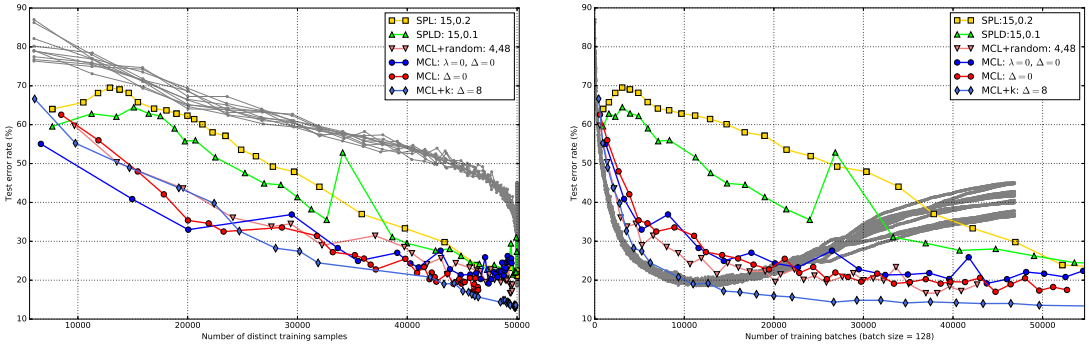

Figure 2: Test error rate (%) vs. number of distinct training sample (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD).

Figure 1-2 show how the test error changes with the number of distinct training samples and the number of training batches, which reflects the training time. The left plot in each figure implies the "sample complexity" of different methods, i.e., how many distinct training samples are needed to achieve the error rate. The right plot shows the convergence rate.

On all datasets, MCL and its variants outperform SPL and SPLD for both sample complexity and convergence rate. They are slightly slower than SGD on convergence but can achieve much smaller error when using the same number of labeled samples. Moreover, when using the same learning rate strategy, they are more robust to overfitting, as shown in Figure 2. In addition, they reduce the error faster than others in later stages, and MCL+k always achieves the lowest test error, as shown in

Table 1. Comparing Figure 1 with Figure 2-3, MCL has significant advantages when applied to deep models.

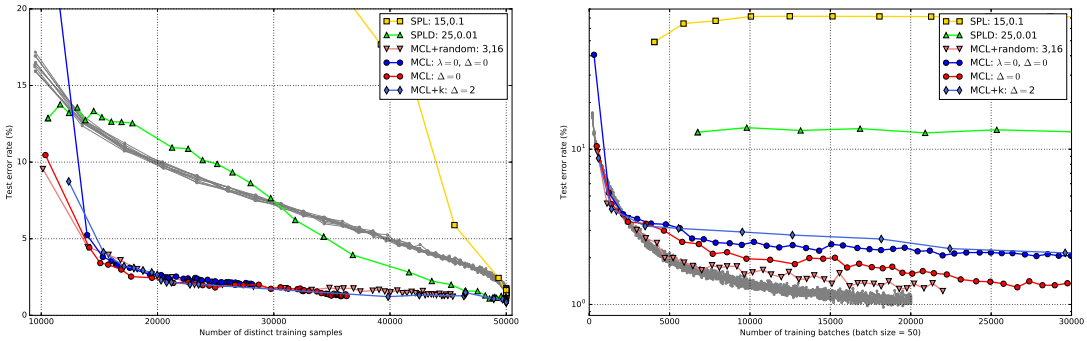

Figure 3: Test error rate (%) vs. number of distinct training sample (left) and number of training batches (right) on MNIST (grey curves represents 10 random trials of SGD).

Among the four variants of MCL, MCL+k achieves the fastest convergence speed and the smallest test error, followed by MCL+random, MCL($\Delta = 0$) and MCL($\lambda = 0, \Delta = 0$) (the only exception is the test error on News20 between the last two variants). This indicates that the diversity introduced by submodular regularization does bring improvement, and changing both hardness and diversity leads to better performance. The combination of MCL and random curriculum speedups the convergence, but still cannot outperform MCL+k. We can obtain similar ranking on sample complexity, but the differences among variants are small.

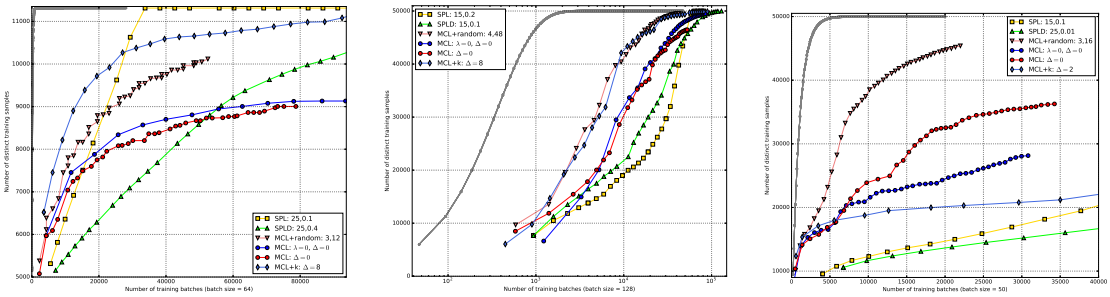

Figure 4: Number of distinct training samples vs. number of training batches for News20 (left), CIFAR10 (middle) and MNIST(right) (grey curves represents 10 random trials of SGD).

Figure 4 shows how the number of distinct training samples changes as training proceeds. It reflects the trade-off between "training on more new samples" vs. "training on fewer samples more often." MCL and its variants usually require much fewer labeled samples than SGD but more than SPL and SPLD. Considering their advantages on the smaller error, better sample complexity, and faster convergence, MCL achieves a promising trade-off.

REFERENCES

Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *ICML*, pp. 1–9, 1998.

E. L. Allgower and Kurt Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, 2003.

Sumit Basu and Janara Christensen. Teaching classification boundaries to humans. In *AAAI*, pp. 109–115, 2013.

Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse m-best solutions in markov random fields. In *ECCV*, pp. 1–16, 2012.

Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear programming - theory and algorithms (2. ed.)*. Wiley, 1993.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Yoshua Bengio. *Evolving Culture Versus Local Minima*, pp. 109–138. Springer Berlin Heidelberg, 2014.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pp. 41–48, 2009.

Ellen Bialystok, Fergus I. M. Craik, and Gigi Luk. Bilingualism: consequences for mind and brain. *Trends in Cognitive Sciences*, 16(4):240–250, 2012.

Michele Conforti and Gerard Cornuejols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984.

Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, pp. 746–751, 2005.

Ido Dagan and Sean P. Engelson. Committee-based sampling for training probabilistic classifiers. In *ICML*, pp. 150–157, 1995.

Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *ICML*, pp. 208–215, 2008.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Farzan Farnia and David Tse. A minimax approach to supervised learning. In *NIPS*, pp. 4240–4248, 2016.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Satoru Fujishige. *Submodular functions and optimization*. Annals of discrete mathematics. Elsevier, 2005.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-optimal map inference for determinantal point processes. In *NIPS*, pp. 2735–2743, 2012.

Rishabh Iyer and Jeff Bilmes. Submodular point processes with applications in machine learning. In *AISTATS*, May 2015.

Rishabh Iyer, Stefanie Jegelka, and Jeff A. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML*, 2013.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander G. Hauptmann. Self-paced learning with diversity. In *NIPS*, pp. 2078–2086, 2014.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. Self-paced curriculum learning. In *AAAI*, pp. 2694–2700, 2015.

Faisal Khan, Bilge Mutlu, and Xiaojin Zhu. How do humans teach: On curriculum learning and teaching dimension. In *NIPS*, pp. 1449–1457, 2011a.

Faisal Khan, Xiaojin (Jerry) Zhu, and Bilge Mutlu. How do humans teach: On curriculum learning and teaching dimension. In *NIPS*, pp. 1449–1457, 2011b.

Ágnes Melinda Kovács and Jacques Mehler. Flexible learning of multiple speech structures in bilingual infants. *Science*, 325(5940):611–612, 2009.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, pp. 1189–1197, 2010.

Gert R.G. Lanckriet, Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research (JMLR)*, 3:555–582, 2003.

Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, pp. 331–339, 1995.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *SIGKDD*, pp. 420–429, 2007.

Ping Li, Jennifer Legault, and Kaitlyn A. Litcofsky. Neuroplasticity as a function of second language learning: Anatomical changes in the human brain. *Cortex*, 58:301–324, 2014.

Andrea Mechelli, Jenny T. Crinion, Uta Noppeney, John O'Doherty, John Ashburner, Richard S. Frackowiak, and Cathy J. Price. Neurolinguistics: Structural plasticity in the bilingual brain. *Nature*, 431(7010):757–757, 2004.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, chapter 27, pp. 234–243. Springer Berlin Heidelberg, 1978.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 1812–1818, 2015.

Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2006.

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.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1): 127–152, 2005.

Kaustubh R Patil, Xiaojin Zhu, Ł ukasz Kopeć, and Bradley C Love. Optimal teaching for limited-capacity human learners. In *NIPS*, pp. 2465–2473, 2014.

Adarsh Prasad, Stefanie Jegelka, and Dhruv Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *NIPS*, pp. 2645–2653, 2014.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *CAIDA*, pp. 309–318, 2001.

Burr Settles. Active learning literature survey. Technical report, University of Wisconsin, Madison, 2010.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT*, pp. 287–294, 1992.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *NIPS 2009 Workshop on Grammar Induction, Representation of Language and Language Learning*, 2009.

James Steven Supancic III and Deva Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pp. 2379–2386, 2013.

Kevin Tang, Vignesh Ramanathan, Li Fei-fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In *NIPS*, pp. 638–646, 2012a.

Ye Tang, Yu-Bin Yang, and Yang Gao. Self-paced dictionary learning for image classification. In *MM*, pp. 833–836, 2012b.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *ICML*, 2014a.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris D. Bartels, and Jeff A. Bilmes. Submodular subset selection for large-scale speech training data. In *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP) 2014*, pp. 3311–3315, 2014b.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *ICML*, 2015.

Tianyi Zhou, Hua Ouyang, Jeff Bilmes, Yi Chang, and Carlos Guestrin. Scaling submodular maximization via pruned submodularity graphs. In *AISTATS*, 2017.

Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, pp. 4083–4087, 2015.

4 APPENDIX

4.1 PROOF OF PROPOSITION 1

Proof. Let $g(x) = \max_i g_i(x)$, where $g_i(x)$ is β-strongly convex for any i. According to a definition of strongly convex function given in Theorem 2.1.8 of (Nesterov, 2004), let $\lambda \in [0, 1]$, we have

$$g_i(\lambda x + (1-\lambda)y) \leq \lambda g_i(x) + (1-\lambda)g_i(y) - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2, \forall i.$$

The following proves that $g(x)$ is also β-strongly convex:

$$\begin{aligned} g(\lambda x + (1-\lambda)y) &= \max_i g_i(\lambda x + (1-\lambda)y) \\ &\leq \max_i \left[\lambda g_i(x) + (1-\lambda)g_i(y)\right] - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2 \\ &\leq \max_i \lambda g_i(x) + \max_i (1-\lambda)g_i(y) - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2 \\ &= \lambda g(x) + (1-\lambda)g(y) - \frac{\beta}{2}\lambda(1-\lambda)\|x-y\|_2^2. \end{aligned}$$

□

4.2 PROOF OF THEOREM 1

Proof. The objective $g(w)$ of the minimax problem in Eq. (2) after eliminating A is given in Eq. (4). Since $G(A)$ in Eq. (5) is monotone non-decreasing submodular, the optimal subset A when defining $g(w)$ in Eq. (4) always has size k if $|V| \geq k$. In addition, because the loss function $L\left(y_i, f(x_i, w)\right)$ is β-strongly convex, $g(w)$ in Eq. (4) is the maximum over multiple $k\beta$-strongly convex functions with different A. According to Proposition 1, $g(w)$ is also $k\beta$-strongly convex, i.e.,

$$g(\hat{w}) \geq g(w^*) + \nabla g(w^*)^T(\hat{w} - w^*) + \frac{k\beta}{2}\|\hat{w} - w^*\|_2^2, \ \ \forall \nabla g(w^*) \in \partial g(w^*). \tag{20}$$

Since the convex function $g(w)$ achieves minimum on w^*, it is valid to substitute $\nabla g(w^*) = 0 \in \partial g(w^*)$ into Eq. (20). After rearrangement, we have

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2}{k\beta}\left[g(\hat{w}) - g(w^*)\right]. \tag{21}$$

In the following, we will prove $g(w^*) \geq \alpha \cdot g(\hat{w})$, which together with Eq. (21) will lead to the final bound showing how close $\hat{w}$ is to w^*.

Note $\hat{g}(w)$ (Eq. (6)) is a piecewise function, each piece of whom is convex and associated with different $\hat{A}$ achieved by a submodular maximization algorithm of approximation factor α. Since $\hat{A}$ is not guaranteed to be a global maxima, unlike $g(w)$, the whole $\hat{g}(w)$ cannot be written as the maximum of multiple convex functions and thus can be non-convex. Therefore, gradient descent in Step 6-9 of Algorithm 1 can lead to either 1) $\hat{w}$ is a global minima of $\hat{g}(w)$; or 2) $\hat{w}$ is a local minima of $\hat{g}(w)$. Saddle points and local maxima do not exist on $\hat{g}(w)$ because each piece of it is convex.

1) When $\hat{w}$ is a global minima of $\hat{g}(w)$, we have

$$g(w^*) \geq \hat{g}(w^*) \geq \hat{g}(\hat{w}) \geq \alpha \cdot g(\hat{w}). \tag{22}$$

The first inequality is due to $g(\cdot) \geq \hat{g}(\cdot)$. The second inequality is due to the global optimality of $\hat{w}$. The third inequality is due to the approximation bound $\hat{g}(\cdot) \geq \alpha \cdot g(\cdot)$ guaranteed by the submodular maximization in Step 7 of Algorithm 1.

2) When $\hat{w}$ is a local minima of $\hat{g}(w)$, we have $\nabla \hat{g}(\hat{w}) = 0$. Let $h(w)$ to be the piece of $\hat{g}(w)$ where $\hat{w}$ is located, then $\hat{w}$ has to be a global minima of $h(w)$ due to the convexity of $h(w)$. Let $\mathcal{A}$ denote the ground set of $\hat{A}$ on all pieces of $\hat{g}(w)$, we define an auxiliary convex function $\tilde{g}(w)$ as

$$\tilde{g}(w) \triangleq \max_{A \in \mathcal{A}} \sum_{i \in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A). \tag{23}$$

It is convex because it is defined as the maximum of multiple convex function. So we have

$$\hat{g}(w) \leq \tilde{g}(w) \leq g(w), \forall w \in \mathbb{R}^m. \tag{24}$$

The first inequality is due to the definition of $\mathcal{A}$, and the second inequality is a result of $\mathcal{A} \subseteq V$ by comparing $g(w)$ in Eq. (4) with $\tilde{g}(w)$ in Eq. (23). Let $\tilde{w}$ denote a global minima of $\tilde{g}(w)$, we have

$$g(w^*) \geq \tilde{g}(w^*) \geq \tilde{g}(\tilde{w}) \geq h(\tilde{w}) \geq h(\hat{w}) = \hat{g}(\hat{w}) \geq \alpha \cdot g(\hat{w}). \tag{25}$$

The first inequality is due to Eq. (24), the second inequality is due to the global optimality of $\tilde{w}$ on $\tilde{g}(w)$, the third inequality is due to the definition of $\tilde{g}(w)$ in Eq. (23) ($\tilde{g}(w)$ is the maximum of all pieces of $\hat{g}(w)$ and $h(w)$ is one piece of them), the fourth inequality is due to the global optimaliy of $\hat{w}$ on $h(w)$, the last inequality is due to the approximation bound $\hat{g}(\cdot) \geq \alpha \cdot g(\cdot)$ guaranteed by the submodular maximization in Step 7 of Algorithm 1.

Therefore, in both cases we have $g(w^*) \geq \alpha \cdot g(\hat{w})$. Applying it to Eq. (21) results in

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2}{k\beta}\left(\frac{1}{\alpha} - 1\right) \cdot g(w^*). \tag{26}$$

□

4.3 Proposition 2

Proposition 2. *If $x \in [0, 1]$, the following inequality holds true.*

$$\frac{x}{1 - e^{-x}} - 1 \leq x. \tag{27}$$

Proof. Due to two inequalities $e^x \leq 1 + x + x^2/2$ for $x \leq 0$ and $1 - e^{-x} \geq x/2$ for $x \in [0, 1]$,

$$\frac{x}{1 - e^{-x}} - 1 = \frac{x - 1 + e^{-x}}{1 - e^{-x}} \leq \frac{x - 1 + (1 - x + x^2/2)}{x/2} = x. \tag{28}$$

□

4.4 Proof of Theorem 2

Proof. Applying the inequality in Proposition 2 and the approximation factor of lazy greedy $\alpha = (1 - e^{-\kappa_G})/\kappa_G$ to the right hand side of Eq. (10) from Theorem 1 yields

$$\begin{aligned} \|\hat{w} - w^*\|_2^2 &\leq \frac{2}{k\beta}\left(\frac{1}{\alpha} - 1\right) \cdot g(w^*) \\ &= \frac{2}{k\beta}\left(\frac{\kappa_G}{1 - e^{-\kappa_G}} - 1\right) \cdot g(w^*) \leq \frac{2\kappa_G}{k\beta} \cdot g(w^*), \end{aligned} \tag{29}$$

where κ_G is the curvature of submodular function $G(\cdot)$ defined in Eq. (5). According to the definition of curvature given in Eq. (9), and let κ_F denotes the curvature of submodular function $F(\cdot)$ in Eq. (5), we have

$$\begin{aligned} \kappa_G &= 1 - \min_{j \in V} \frac{L(j) + \lambda F(j|V \backslash j)}{L(j) + \lambda F(j)} = \lambda \cdot \max_{j \in V} \frac{F(j) - F(j|V \backslash j)}{L(j) + \lambda F(j)} \\ &= \lambda \cdot \max_{j \in V} \frac{1 - \frac{F(j|V \backslash j)}{F(j)}}{\frac{L(j)}{F(j)} + \lambda} \leq \frac{\lambda \cdot \kappa_F}{\min_{j \in V} \frac{L(j)}{F(j)} + \lambda} = \frac{\kappa_F}{c_1/\lambda + 1}, \quad c_1 \triangleq \min_{j \in V, t} \frac{L(j, t)}{F(j)} \end{aligned} \tag{30}$$

where we use $L(j, t)$ as a simpler representation of per-sample loss $L(y_j, f(x_i, \hat{w}^t))$ if not causing any confusion, and the last inequality is due to the definition of curvature κ_F for submodular function $F(\cdot)$. Substituting the above inequality about κ_G into Eq. (29) results in

$$\|\hat{w} - w^*\|_2^2 \leq \frac{2\kappa_F}{k\beta(c_1/\lambda + 1)} \leq \frac{2\kappa_F}{\beta c_1} \times \frac{\lambda}{k} \times g(w^*) \tag{31}$$

We use subscript as the index for iterations in the outer-loop, e.g., $\hat{w}_T$ is the model weights w after the Tth iteration of outer-loop. If we decrease λ exponentially from $\lambda = \lambda_0$ and increase k linearly from $k = k_0$, as Step 11 in Algorithm 1, we have

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1} \times \frac{(1 - \gamma)^T}{(k_0 + T\Delta)} \times g(w_T^*), \tag{32}$$

According to the definition of $g(\cdot)$ in Eq. (4), for $g(w_T^*)$ we have

$$\begin{aligned} g(w_T^*) &= \min_{w\in\mathbb{R}^m} \max_{A\subseteq V, |A|\leq k} \sum_{i\in A} L\left(y_i, f(x_i, w)\right) + \lambda F(A) \\ &\leq \min_{w\in\mathbb{R}^m} \max_{A\subseteq V, |A|\leq k} \sum_{i\in A} L\left(y_i, f(x_i, w)\right) + \lambda_0 (1-\gamma)^T \max_{A\subseteq V, |A|\leq k} F(A) \\ &= g(w_\infty^*) + \lambda_0 (1-\gamma)^T c_2, \end{aligned} \tag{33}$$

where

$$g(w_\infty^*) \triangleq \min_{w\in\mathbb{R}^m} \max_{A\subseteq V, |A|\leq k} \sum_{i\in A} L\left(y_i, f(x_i, w)\right), c_2 \triangleq \max_{A\subseteq V, |A|\leq k} F(A). \tag{34}$$

Substituting Eq. (33) to Eq. (32) yields

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1} \times \frac{(1-\gamma)^T}{(k_0 + T\Delta)} \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^T\right], \tag{35}$$

If we can tolerate more expensive computational cost for running submodular maximization with larger budget k, and increase k exponentially, i.e., $k \leftarrow (1+\Delta)\cdot k$, we have

$$\|\hat{w}_T - w_T^*\|_2^2 \leq \frac{2\kappa_F \lambda_0}{\beta c_1 k_0} \times \left(\frac{1-\gamma}{1+\Delta}\right)^T \times \left[g(w_\infty^*) + \lambda_0 c_2 (1-\gamma)^T\right]. \tag{36}$$

This completes the proof. □

4.5 Proposition 3

Proposition 3. *If $x \in [0,1]$, the following inequality holds true.*

$$\frac{x}{1-e^{-x}} - \frac{1-e^{-x}}{x} \leq \frac{3}{2}x. \tag{37}$$

Proof. Proposition 2 leads to

$$\frac{x}{1-e^{-x}} - \frac{1-e^{-x}}{x} \leq x + 1 - \frac{1-e^{-x}}{x} = \frac{x^2 + x - 1 + e^{-x}}{x}. \tag{38}$$

By substituting inequality $e^x \leq 1 + x + x^2/2$ for $x \leq 0$ into above inequality, we have

$$\frac{x^2 + x - 1 + e^{-x}}{x} = \frac{x^2 + x - 1 + (1 - x + x^2/2)}{x} \leq \frac{3}{2}x. \tag{39}$$

□

4.6 Proof of Lemma 1

Proof. On the optimization path from $\hat{w}^p$ to $\hat{w}$, $\hat{g}(\cdot)$ may switch to different convex pieces. Hence, we use $h^t(\cdot)$ to denote the piece of $\hat{g}(\cdot)$ on which $\hat{w}^t$ is located, so $\hat{g}(\hat{w}^t) = h^t(\hat{w}^t)$. Because $h^t(\cdot)$ is $k\beta$-strongly convex,

$$\begin{aligned} \|\hat{w}^{t+1} - \hat{w}\|^2 &= \|\hat{w}^t - \eta\nabla h^t(w^t) - \hat{w}\|^2 \\ &= \|\hat{w}^t - \hat{w}\|^2 - 2\eta\nabla h^t(w^t)\cdot(\hat{w}^t - \hat{w}) + \eta^2\|\nabla h^t(\hat{w}^t)\|^2 \\ &\leq \|\hat{w}^t - \hat{w}\|^2 - 2\eta\left(h^t(\hat{w}^t) - h^t(\hat{w}) + \frac{k\beta}{2}\|\hat{w}^t - \hat{w}\|^2\right) + \eta^2\|\nabla h^t(\hat{w}^t)\|^2 \\ &\leq (1 - k\beta\eta)\|\hat{w}^t - \hat{w}\|^2 - 2\eta\left(h^t(\hat{w}^t) - h^t(\hat{w})\right) + \eta^2\|\nabla h^t(\hat{w}^t)\|^2. \end{aligned} \tag{40}$$

Recall the auxiliary convex function $\tilde{g}(\cdot)$ defined in Eq. (23), by using the inequality shown in Eq. (25), we have

$$\begin{aligned} \hat{g}(\hat{w}) \leq \tilde{g}(\tilde{w}) &\leq \tilde{g}\left(w - \eta\nabla\hat{g}(w)\right) \leq \frac{1}{\alpha}\hat{g}\left(w - \eta\nabla\hat{g}(w)\right) \\ &\leq \frac{1}{\alpha}\left[\hat{g}(w) - \eta\|\nabla\hat{g}(w)\|^2 + \frac{\eta^2 kL}{2}\|\nabla\hat{g}(w)\|^2\right] \\ &= \frac{1}{\alpha}\hat{g}(w) - \frac{\eta}{\alpha}\left(1 - \frac{\eta kL}{2}\right)\|\nabla\hat{g}(w)\|^2, \end{aligned} \tag{41}$$

where the third inequality is due to the approximation factor α, and the fourth inequality is due to the kL-smoothness of $\hat{g}(\cdot)$. Eq. (41) results in

$$\|\nabla \hat{g}(w)\|^2 \leq \frac{2\alpha}{\eta(2-\eta kL)}\left[\frac{1}{\alpha}\hat{g}(w) - \hat{g}(\hat{w})\right]. \tag{42}$$

Let $w = \hat{w}^t$ and substitute Eq. (42) to Eq. (40), due to $\hat{g}(\hat{w}^t) = h^t(\hat{w}^t)$, we have

$$\begin{aligned}
\|\hat{w}^{t+1} - \hat{w}\|^2 &\leq (1-k\beta\eta)\|\hat{w}^t - \hat{w}\|^2 - 2\eta\left(h^t(\hat{w}^t) - h^t(\hat{w})\right) + \eta^2\|\nabla h^t(\hat{w}^t)\|^2 \\
&\leq (1-k\beta\eta)\|\hat{w}^t - \hat{w}\|^2 - 2\eta\left(\hat{g}(\hat{w}^t) - \frac{1}{\alpha}\hat{g}(\hat{w})\right) + \frac{2\eta\alpha}{2-\eta kL}\left[\frac{1}{\alpha}\hat{g}(\hat{w}^t) - \hat{g}(\hat{w})\right] \\
&= (1-k\beta\eta)\|\hat{w}^t - \hat{w}\|^2 + 2\eta\left[\left(\frac{1}{2-\eta kL} - 1\right)\hat{g}(\hat{w}^t) + \left(\frac{1}{\alpha} - \frac{\alpha}{2-\eta kL}\right)\hat{g}(\hat{w})\right] \\
&= \left(1 - \frac{\beta}{L}\right)\|\hat{w}^t - \hat{w}\|^2 + \frac{2}{kL}\left(\frac{1}{\alpha} - \alpha\right)\hat{g}(\hat{w}).
\end{aligned} \tag{43}$$

The last step is achieved by setting learning rate $\eta = 1/(kL)$. Recursively applying the above inequality from $t = p-1$ to $t = 1$ (p is the number of iterations for inner loop) yields

$$\|\hat{w}^p - \hat{w}\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \|\hat{w}^0 - \hat{w}\|^2 + \frac{2\left[1 - \left(1 - \frac{\beta}{L}\right)^{p+1}\right]}{k\beta} \cdot \left(\frac{1}{\alpha} - \alpha\right) \cdot \hat{g}(\hat{w}). \tag{44}$$

Since $\alpha = \frac{1-e^{-\kappa_G}}{\kappa_G}$, Proposition 3 leads to

$$\frac{1}{\alpha} - \alpha \leq \frac{3}{2}\kappa_G \leq \frac{3\kappa_F}{2c_1/\lambda + 2}, \tag{45}$$

where the last inequality comes from Eq. (30). In addition, due to inequality $\left(1 + \frac{x}{n}\right)^n \geq 1 + x$ for $n > 1$ and $|x| \leq n$, we have

$$1 - \left(1 - \frac{\beta}{L}\right)^{p+1} \leq \frac{\beta(p+1)}{L}. \tag{46}$$

Applying Eq. (45) and Eq. (46) to Eq. (44) results in

$$\|\hat{w}^p - \hat{w}\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \|\hat{w}^0 - \hat{w}\|^2 + \frac{3\kappa_F(p+1)}{kL(c_1/\lambda + 1)} \cdot \hat{g}(\hat{w}). \tag{47}$$

This completes the proof. □

4.7 Proof of Theorem 3

Proof. Triangle inequality results in

$$\|\hat{w}_T^p - w_{\mathcal{T}}^*\|^2 \leq 2\left(\|\hat{w}_T^p - \hat{w}_T\|^2 + \|\hat{w}_T - w_{\mathcal{T}}^*\|^2 + \|\hat{w}_{\mathcal{T}}^* - w_{\mathcal{T}}^*\|^2\right) \tag{48}$$

In the right hand side of above inequality, the first term can be upper bounded by the result in Lemma 1, while the second term can be upper bounded by the results in Theorem 2. Now we start to study an upper bound for the third term. Since the third term has two different subscript representing different iterations for the outer-loop, we use $g_T(\cdot)$ to represent $g(\cdot)$ (defined in Eq. (4)) corresponding to the Tth iteration for the outer loop, i.e., when $\lambda = \lambda_0(1-\gamma)^T$.

$$\begin{aligned}
\|w_T^* - w_{\mathcal{T}}^*\|^2 &\leq \frac{2}{k\beta}\left(g_{\mathcal{T}}(w_T^*) - g_{\mathcal{T}}(w_{\mathcal{T}}^*)\right) \\
&\leq \frac{2}{k\beta}\left(g_T(w_T^*) - g_{\mathcal{T}}(w_{\mathcal{T}}^*)\right) \\
&\leq \frac{2}{k\beta}\left(g_T(w_{\mathcal{T}}^*) - g_{\mathcal{T}}(w_{\mathcal{T}}^*)\right),
\end{aligned} \tag{49}$$

where the first inequality is due to $k\beta$-strong convexity of $g_{\mathcal{T}}(\cdot)$ and the global optimality of $w_{\mathcal{T}}^*$ w.r.t. $g_{\mathcal{T}}(\cdot)$, the second inequality is due to $g_T(w) \geq g_{\mathcal{T}}(w)$ for any w given $\mathcal{T} \geq T$, the third inequality is due to the global optimality of w_T^* w.r.t. $g_T(\cdot)$. Let A_T and $A_{\mathcal{T}}$ be the optimal A achieving the

maximum of $G(A)$ (defined in Eq. (5)) when $w = w_T^*$ and $w = w_{\mathcal{T}}^*$ respectively, then

$$g_T(w_{\mathcal{T}}^*) = \sum_{i \in A_T} L\left(y_i, f(x_i, w_{\mathcal{T}}^*)\right) + \lambda_0 (1-\gamma)^T F(A_T), \tag{50}$$

$$g_{\mathcal{T}}(w_{\mathcal{T}}^*) = \sum_{i \in A_{\mathcal{T}}} L\left(y_i, f(x_i, w_{\mathcal{T}}^*)\right) + \lambda_0 (1-\gamma)^{\mathcal{T}} F(A_{\mathcal{T}}). \tag{51}$$

Due to the optimality of $A_{\mathcal{T}}$, we have

$$\begin{aligned} g_T(w_{\mathcal{T}}^*) - g_{\mathcal{T}}(w_{\mathcal{T}}^*) \leq & \sum_{i \in A_T} L\left(y_i, f(x_i, w_{\mathcal{T}}^*)\right) + \lambda_0 (1-\gamma)^T F(A_T) - \\ & \sum_{i \in A_T} L\left(y_i, f(x_i, w_{\mathcal{T}}^*)\right) + \lambda_0 (1-\gamma)^{\mathcal{T}} F(A_T) \\ = & \lambda_0 (1-\gamma)^T \left[1 - (1-\gamma)^{\mathcal{T}-T}\right] F(A_T) \\ \leq & \lambda_0 \gamma (\mathcal{T} - T)(1-\gamma)^T F(A_T), \end{aligned} \tag{52}$$

where the last inequality is a result of applying inequality $\left(1 + \frac{x}{n}\right)^n \geq 1 + x$ for $n > 1$ and $|x| \leq n$. Applying Eq. (52) to Eq. (49) gives us an upper bound for the third term, i.e.,

$$\|w_T^* - w_{\mathcal{T}}^*\|^2 \leq \frac{2\lambda_0 \gamma (\mathcal{T} - T)(1-\gamma)^T}{k\beta} \cdot F(A_T). \tag{53}$$

By combining Eq. (14) from Lemma 1 giving an upper bound for the first term in Eq. (48), Eq. (11) from Theorem 2 giving an upper bound for the second term, and Eq. (53) giving an upper bound for the third term, after some re-arrangements, we have the following result.

$$\|\hat{w}_T^p - w_{\mathcal{T}}^*\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \cdot 2\|\hat{w}_T^0 - \hat{w}_T\|^2 + \frac{\lambda}{k} \cdot 2c, \tag{54}$$

when λ starts $\lambda = \lambda_0$ and decreases exponentially, i.e., $\lambda \leftarrow (1-\gamma) \cdot \lambda$, c is defined as

$$c = \frac{\kappa_F}{c_1} \cdot \left[\frac{3(p+1)}{L} \cdot \hat{g}(\hat{w}_T) + \frac{2}{\beta} \cdot g(w_\infty^*) + \frac{2c_2 \lambda_0 (1-\gamma)^T}{\beta}\right] + \frac{2\gamma(\mathcal{T} - T)}{\beta} \cdot F(A_T). \tag{55}$$

By using the fact $F(A_T) \leq c_2 = \max_{A \subseteq V, |A| \leq k} F(A)$, c can be further replaced by its upper bound

$$c \leq \frac{\kappa_F}{c_1} \cdot \left[\frac{3(p+1)}{L} \cdot \hat{g}(\hat{w}_T) + \frac{2}{\beta} \cdot g(w_\infty^*)\right] + \frac{2\kappa_F \lambda_0 (1-\gamma)^T + 2c_1 \gamma (\mathcal{T} - T)}{c_1 \beta} \cdot c_2. \tag{56}$$

In addition, Eq. (22) and Eq. (25) show in both cases $\hat{g}(\hat{w}_T) \leq g(w_T^*)$, by using the inequality in Eq. (33), we have

$$\hat{g}(\hat{w}_T) \leq g(w_T^*) \leq g(w_\infty^*) + \lambda_0 (1-\gamma)^T c_2. \tag{57}$$

Hence, the first term in Eq. (56) can be further simplified by using the above inequality, so we have

$$c \leq \frac{\kappa_F}{c_1} \left[\frac{3(p+1)}{L} + \frac{2}{\beta}\right] \cdot g(w_\infty^*) + \left[\frac{\kappa_F}{c_1} \left(\frac{3(p+1)}{L} + \frac{2}{\beta}\right) \lambda_0 (1-\gamma)^T + \frac{2\gamma(\mathcal{T} - T)}{\beta}\right] \cdot c_2. \tag{58}$$

If we increase k linearly from $k = k_0$ as Step 11 in Algorithm 1, we have

$$\|\hat{w}_T^p - w_{\mathcal{T}}^*\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \cdot 2\|\hat{w}_T^0 - \hat{w}_T\|^2 + \frac{(1-\gamma)^T}{k_0 + T\Delta} \cdot 2c\lambda_0. \tag{59}$$

If we can tolerate more expensive computational cost for running submodular maximization with larger budget k, and increase k exponentially, i.e., $k \leftarrow (1+\Delta) \cdot k$, we instead have

$$\|\hat{w}_T^p - w_{\mathcal{T}}^*\|^2 \leq \left(1 - \frac{\beta}{L}\right)^p \cdot 2\|\hat{w}_T^0 - \hat{w}_T\|^2 + \left(\frac{1-\gamma}{1+\Delta}\right)^T \cdot \frac{2c\lambda_0}{k_0}. \tag{60}$$

This completes the proof. □

4.8 Submodular Maximization used in Our Implementation

In our implementation, we use two strategies to accelerate the computation: 1) reduce the size of ground set by pruning methods before running any submodular maximization algorithm, we use (Wei et al., 2014a) here for its simplicity but running (Zhou et al., 2017) afterwards can reduce additional elements; 2) update A by a subgradient ascent step (which costs $|V|$ function evaluations) of majorization-minimization (MM) scheme (Iyer et al., 2013) when a targeted approximation factor α has already been achieved on $\hat{A}$ from the previous iteration, otherwise the lazy greedy algorithm is applied.

Algorithm 2 SUBMODUARMAX($G, k, \hat{A}, \alpha \in [0, 1)$)

1: **input:** $G(\cdot)$, k, $\hat{A}$, α
2: **output:** $\hat{A}$
3: Reduce ground set V: arrange V non-increasingly in terms of $G(i|V\backslash i)$ in a permutation δ where $\delta(k)$ is the kth element, $V^{'} \leftarrow \{i \in V | G(i) \geq G(\delta(k)|V\backslash\delta(k))\}$;
4: Compute upper bound to maximum of Eq. (5):

$$\tau = \max_{A \in V, |A| \leq k} \sum_{i \in A} \left[L\left(y_i, f(x_i, w^t)\right) + \lambda F(i) \right]$$

5: **if** $G(\hat{A}) \geq \alpha \cdot \tau$ **then**
6: Permutation σ of $V^{'}$: the first k elements $S_k^\sigma = \hat{A}$ with random order, the rest elements in σ with non-decreasing $G(i|V^{'}\backslash i)/G(i)$;
7: Define modular function $h_{\hat{A}}(A) \triangleq \sum_{i \in A} h_{\hat{A}}(i)$ and compute $h_{\hat{A}}(\sigma(i)) = G(S_i^\sigma) - G(S_{i-1}^\sigma)$;
8: Compute lower bound $L(A)$ of $G(A)$:

$$L(A) \triangleq G(\hat{A}) + h_{\hat{A}}(A) - h_{\hat{A}}(\hat{A})$$

9: $\hat{A} \leftarrow \text{argmax}_{A \in V', |A| \leq k} L(A)$;
10: **else**
11: $\hat{A} \leftarrow LazyGreedy(G, V^{'}, k)$;
12: **end if**

The complete algorithm is given in Algorithm 2. Step 3 denotes the pruning procedure given in (Wei et al., 2014a). Step 4 compute an upper bound τ for $\max_{A \in V, |A| \leq k} G(A)$ due to submodularity. Step 5-12 performs submodular maximization: if the targeted approximation factor α is already achieved on $\hat{A}$ we run subgradient ascent to further improve it, otherwise a lazy greedy algorithm is applied as in Step 11. Note the modular maximization in Step 4 and Step 9 simply needs to find the top k elements from $V^{'}$ with the largest objective value. Step 6-9 is identical to one step of MM scheme (Iyer et al., 2013) except Step 6 which arranges the elements $V^{'}\backslash\hat{A}$ in a non-decreasing order of $G(i|V^{'}\backslash i)/G(i)$. The set of the first i elements in permutation σ is denoted by S_i^σ in Step 7. Step 6 places fewer elements before the ones with small $G(i|V^{'}\backslash i)/G(i)$ in σ, and intuitively tends to reduce the gap between $L(A)$ and $G(A)$ for any subset A. To see this, let $A_{i-1}^\sigma \triangleq \{\sigma(j) \in A | j < i\}$ for any $i \in A$, the gap we want to reduce is

$$G(A) - L(A) = \sum_{\sigma(i) \in A} G(\sigma(i)) \cdot \left[\frac{G(\sigma(i)|A_{i-1}^\sigma)}{G(\sigma(i))} - \frac{h_{\hat{A}}(\sigma(i))}{G(\sigma(i))} \right], \tag{61}$$

where the second term in the brackets is

$$\frac{h_{\hat{A}}(\sigma(i))}{G(\sigma(i))} = \frac{G(\sigma(i)|S_{i-1}^\sigma)}{G(\sigma(i))} \geq \frac{G(\sigma(i)|V^{'}\backslash\sigma(i))}{G(\sigma(i))}. \tag{62}$$

The elements $\sigma(i)$ with small rank i associates with small S_{i-1}^σ, which often results in (but does not guarantee) small $h_{\hat{A}}(\sigma(i))/G(\sigma(i))$ due to submodularity. According to Step 6, the elements with large rank in σ has large $G(\sigma(i)|V^{'}\backslash\sigma(i))/G(\sigma(i))$, the lower bound for $h_{\hat{A}}(\sigma(i))/G(\sigma(i))$ in Eq. (62). Together they indicate large $h_{\hat{A}}(\sigma(i))/G(\sigma(i))$ and thus a small gap in Eq. (61).

It has been proved in (Iyer et al., 2013) that Step 6-9 produces a non-decreasing $G(\hat{A})$, i.e., if $\hat{A}^+$ is the optimal $\hat{A}$ achieved in Line 9, we have

$$G(\hat{A}^+) \geq L(\hat{A}^+) \geq L(\hat{A}) = G(\hat{A}). \tag{63}$$

The first inequality is due to that $L(\cdot)$ is the lower bound of $G(A)$; the second inequality is due to the optimality of $\hat{A}^+$; the equality is due to $S_k^\sigma = \hat{A}$.

Lemma 2. *Algorithm 2 outputs a solution $\hat{A}$ such that $G(\hat{A}) \geq \min\{\alpha, \alpha_{LG}\} \cdot \max_{A \in V, |A| \leq k} G(A)$, where α_{LG} is the approximation factor of lazy greedy.*

Proof. Firstly, let A^* denote the optimal solution to Eq. (5):

$$A^* \triangleq \text{argmax}_{A \in V, |A| \leq k} \sum_{i \in A} L\left(y_i, f(x_i, w^t)\right) + \lambda F(A). \tag{64}$$

Dataset	News20	MNIST	CIFAR10
#Training	11314	50000	50000
#Test	7532	10000	10000
#Feature	129791	28×28	32×32
#Class	20	10	10

Table 3: Information of datasets.

Dataset	News20	MNIST	CIFAR10
p	50	50	20
#cluster	200	1000	1000
γ	0.05	0.05	0.05
initial k	4	4	4
initial λ	6×10^{-6}	1×10^{-6}	1×10^{-7}
initial η	3.5	0.02	0.01

Table 4: Parameters of MCL (Algorithm 1) and its variants for different datasets.

We prove that τ in Step 4 is an upper bound to $G(A^*)$ as follows.

$$\begin{aligned} \tau &\geq \sum_{i \in A^*} \left[L\left(y_i, f(x_i, w^t)\right) + \lambda F(i) \right] \\ &= \sum_{i \in A^*} L\left(y_i, f(x_i, w^t)\right) + \lambda \sum_{i \in A^*} F(i) \\ &\geq \sum_{i \in A^*} L\left(y_i, f(x_i, w^t)\right) + \lambda F(A^*). \end{aligned} \tag{65}$$

The first inequality is due to the fact that τ is the maximum of the objective in Step 4; the last inequality is due to submodularity, which guarantees $F(i) \geq F(i|B)$ for any $B \subseteq V$.

When $G(\hat{A}) \geq \alpha \cdot \tau$ (Step 5), as we discussed before, the subgradient ascent in Step 6-9 does not decrease the objective, so we have $G(\hat{A}) \geq \alpha \cdot \max_{A \in V, |A| \leq k} G(A)$ for $\hat{A}$ obtained in Step 9. Otherwise, we run lazy greedy on the reduced ground set $V^{'}$. According to Lemma 1 in (Wei et al., 2014a), the ground set reduction in Step 3 does not change the objective value achieved by lazy greedy. So we have $G(\hat{A}) \geq \alpha_{LG} \cdot \max_{A \in V, |A| \leq k} G(A)$ for $\hat{A}$ obtained in Step 11. This completes the proof. □

Algorithm 2 simply combines three proposed submodular maximization techniques, but effectively accelerates the update of $\hat{A}$ in practice. Firstly, as λ decreasing, $G(A)$ turns to be more modular, and $G(i|V \backslash i)$ becomes closer to $G(i)$. So the pruning in Step 3 becomes similar to sorting, and rules out more elements from V. Secondly, $G(A)$ changes smoothly with w^t between two iterations within Algorithm 1. When the learning rate η is small, $\hat{A}$ from the previous iteration can still achieve a large approximation factor, in which case running lazy greedy is not necessary, and Algorithm 2 thus uses faster subgradient ascent to update $\hat{A}$.

This mechanism of setting targeted approximation factor α provides a trade-off between α and computational cost: if α is large, we can obtain large approximation factor but the computation might be expensive since lazy greedy is used more frequently; if α is small, single subgradient ascent will be used more often and the computational cost reduces, but the approximation factor also decrease. Note if we set $\alpha = 1$, Algorithm 2 is forced to run lazy greedy to update $\hat{A}$.

In experiments, we set $\alpha = 0.5$. Though $\alpha = 1$ might lead to smaller approximation error, empirically $\alpha = 0.5$ often results in less computation for Algorithm 2 while maintaining promising performance.

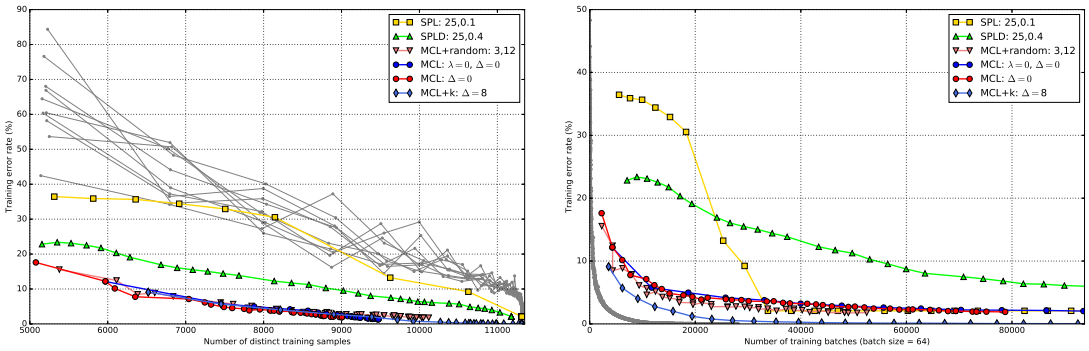

Figure 5: Training error rate (%) vs. number of distinct training sample (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

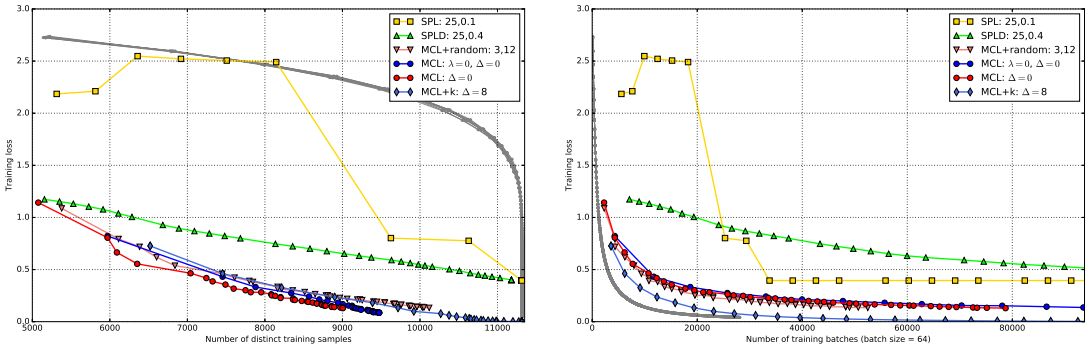

Figure 6: Training loss vs. number of distinct training sample (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

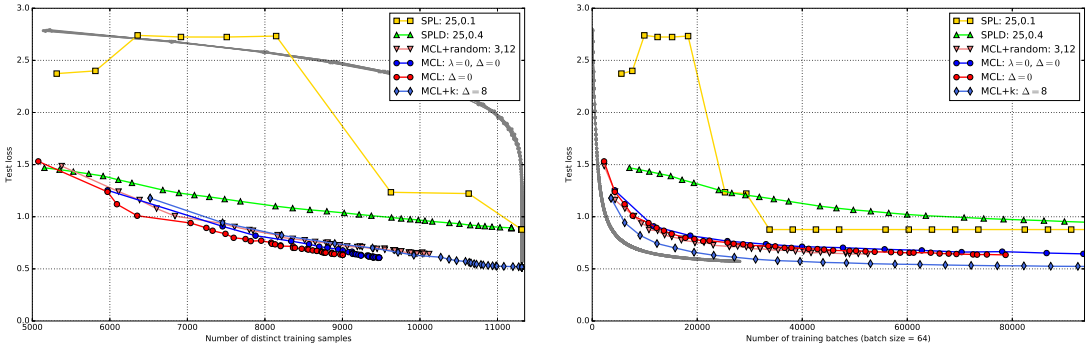

Figure 7: Test loss vs. number of distinct training sample (left) and number of training batches (right) on 20newsgroups (grey curves represents 10 random trials of SGD).

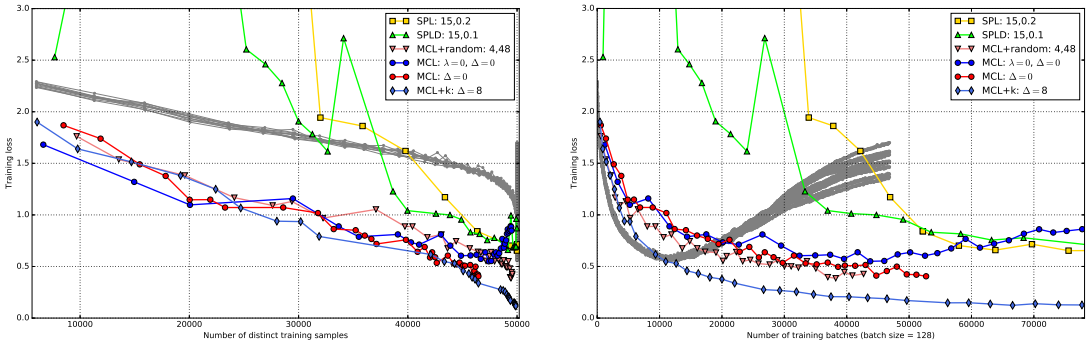

Figure 8: Training loss vs. number of distinct training sample (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD).

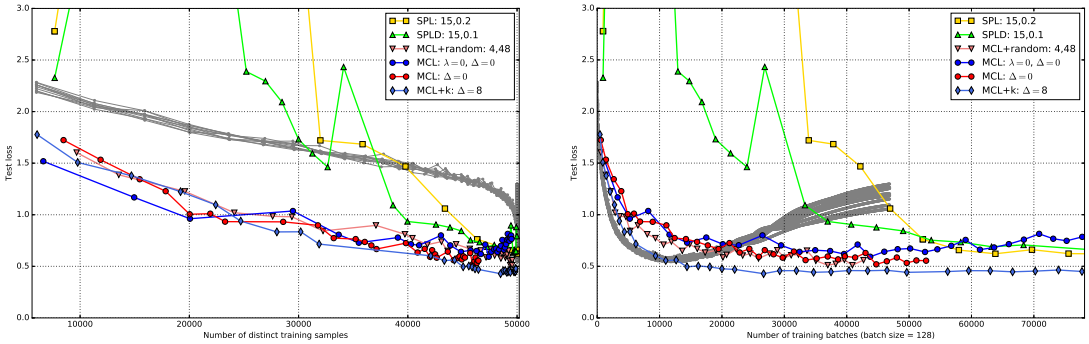

Figure 9: Test loss vs. number of distinct training sample (left) and number of training batches (right) on CIFAR10 (grey curves represents 10 random trials of SGD).

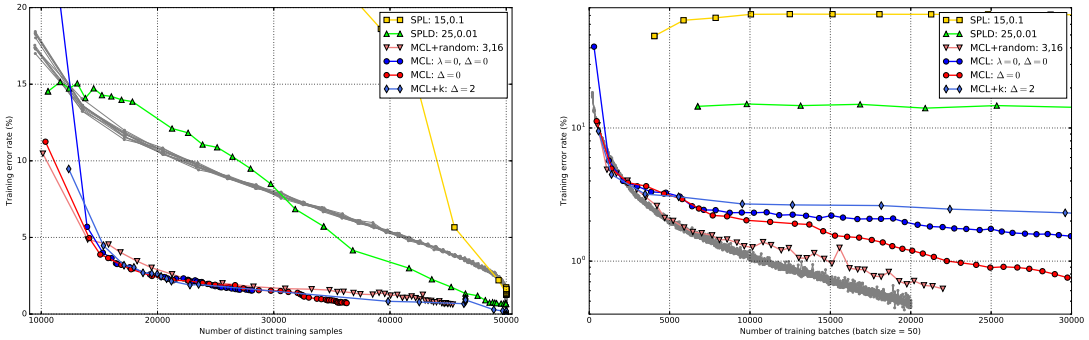

Figure 10: Training error rate (%) vs. number of distinct training sample (left) and number of training batches (right) on MNIST (grey curves represents 10 random trials of SGD).