entropy

MDPI

Article

Classification Active Learning Based on Mutual Information

Jamshid Sourati [1,*], Murat Akcakaya [2], Jennifer G. Dy [1], Todd K. Leen [3,†] and Deniz Erdogmus [1,†]

1 Department of Electrical and Computer Engineering, Northeastern University, 360 Huntington Ave, Boston, MA 02115, USA; jdy@ece.neu.edu (J.G.D.); erdogmus@ece.neu.edu (D.E.)
2 Department of Electrical and Computer Engineering, University of Pittsburgh, 3700 O'Hara Street, Pittsburgh, PA 15261, USA; akcakaya@pitt.edu
3 National Science Foundation, 4201 Wilson Boulevard, Arlington, VA 22230, USA; tleen@nsf.gov
* Correspondence: sourati@ece.neu.edu; Tel.: +1-617-373-4779
† These authors contributed equally to this work.

Academic Editors: Badong Chen and Jose C. Principe
Received: 15 December 2015; Accepted: 1 February 2016 ; Published: 5 February 2016

Abstract: Selecting a subset of samples to label from a large pool of unlabeled data points, such that a sufficiently accurate classifier is obtained using a reasonably small training set is a challenging, yet critical problem. Challenging, since solving this problem includes cumbersome combinatorial computations, and critical, due to the fact that labeling is an expensive and time-consuming task, hence we always aim to minimize the number of required labels. While information theoretical objectives, such as mutual information (MI) between the *labels*, have been successfully used in sequential querying, it is not straightforward to generalize these objectives to batch mode. This is because evaluation and optimization of functions which are trivial in individual querying settings become intractable for many objectives when we are to select multiple queries. In this paper, we develop a framework, where we propose efficient ways of evaluating and maximizing the MI between labels as an objective for batch mode active learning. Our proposed framework efficiently reduces the computational complexity from an order proportional to the batch size, when no approximation is applied, to the linear cost. The performance of this framework is evaluated using data sets from several fields showing that the proposed framework leads to efficient active learning for most of the data sets.

Keywords: active learning; mutual information; submodular maximization; classification

1. Introduction

In supervised learning, there is a teacher in the form of labels associated to a training set of samples to enable learning of models for prediction. However, obtaining expert/human/manual labeling is expensive. Instead of randomly selecting samples for manual annotation (the standard setting), the goal of *active learning* is to intelligently select samples for annotation that enables efficiently learning an accurate classifier with as few labels as possible. Here, the implicit assumptions are that the labeling costs (in terms of time or financial expense) are the same for all of the queries and also significantly larger than the computational cost of the querying algorithms. The latter assumption leads us towards the active learning algorithms that generate smaller batch of queries, even though they might be computationally more expensive compared to the cheap passive learning.

An active learning setting typically starts with an initial model (from a few labeled samples), then more samples are selected for label querying. There are two general strategies for querying: *sequential* or *batch mode*. In sequential querying, a single sample is selected for querying at each active learning step, where each step involves labeling the query and retraining of the model. Batch mode

querying, on the other hand, allows labeling of multiple samples at each active learning step. Most of the classical studies in active learning use sequential querying [1–6]. However, oftentimes querying a batch of samples is more efficient when the experts can label multiple samples in one step, since they can label more queries at each step without the need to wait for the retraining process. In this paper, we introduce a batch mode active learning algorithm based on mutual information.

Performing active learning in batch mode introduces new challenges. Since we need to select a *set* of queries, one should also make sure that the samples are non-redundant to maximize the amount of information that they provide. Another related challenge is that selecting a subset of samples based on a given objective function defined over sets can easily lead to intractable or to non-deterministic polynomial-time hard (NP-hard) optimization problems.

Recently, different batch mode active learning algorithms have been developed. Besides the querying strategy, active learning methods also differ based on the criterion they optimize for selecting queries. [7–9] select samples that reduce model *uncertainty*. Holub *et al.* [8] do this directly using the joint entropy function; Brinker [7] does this indirectly based on the distance of samples to the classifier's boundary; and Chen *et al.* [9] also do this indirectly based on the volume of version space. Most of these methods need to employ heuristics to introduce diversity among the queries. Choosing a subset of samples that maximizes the expected model change [10] is another type of batch selection strategy which works directly with classifier performance, but is usually tied to a particular classification model and is not general. On the other hand, Azimi *et al.* [11] develop a framework which constructs a batch mode variant of any given sequential querying policy, such that it performs close to the sequential scenario. Their algorithm is based on the restrictive assumption that any sequential querying outperforms its batch mode correspondent, in the expense of more frequent model updating. There are also studies that select the queries based on the amount of information they carry with respect to the underlying data distribution. For example, Hoi *et al.* [12] use the Fisher information ratio, which is specifically designed for a logistic regression model; Guo [13] and Li *et al.* [14] utilize mutual information between the input feature values of the candidate queries and the remaining samples, where they used Gaussian Process distribution to model the joint probability over the instances.

In this paper, we advance the field by introducing a general framework for batch mode active learning that selects samples based on mutual information (MI) between *labels* of the candidate query set and the remaining samples given the current model. Our framework is general, because it can be applied with any classifier. Also note that we optimize for MI between the labels; in contrast, the MI in [13] and [14] are based on the input feature values. Our formulation is discriminative; hence, we do not need to model the distribution of the input features. Another additional benefit is that our MI-based objective takes redundancy into account naturally.

MI between the labels has been employed in sequential active learning settings [15]; however, to date, we are not aware of any work that optimizes for this objective directly in batch mode. This is due to two main hurdles: (1) difficulty in calculating the MI between non-singleton subsets of labels and (2) its combinatorial optimization problem. We address hurdle (1) by introducing *pessimistic* and *optimistic* versions of estimating MI, both of which can be calculated efficiently in a greedy fashion (Section 2.2), and hurdle (2) via the popular greedy submodular maximization algorithm and a stochastic version of it. We show that MI is submodular but non-monotone, and therefore only the stochastic algorithm has guaranteed tight bounds for maximizing it (Section 2.3). This stochastic optimization algorithm, which has been not exploited in active learning community before, will be compared in practice against the former algorithm which has been used widely in batch querying [9,12,16,17]. Our proposed framework efficiently reduces the computational complexity from order k (denoting the batch size), when no approximation is applied, to the linear cost (Section 2.4). Additionally, as suggested by Wei *et al.* [18], we also use uncertainty sampling to downsample the unlabeled data as a pre-processing step to further decrease the model complexity

(Section 2.5). The performance of our proposed algorithms are evaluated using data sets from several different fields, and make comparisons against entropy-based and random querying benchmarks.

2. Active Learning with Mutual Information

In this section, we introduce our notations and formulate the batch active learning problem with MI as the objective. Then we provide our solutions to the two hurdles mentioned above.

2.1. Formulation

Let $X = \{\mathbf{x}_i\}_{i=1}^{n+m} \subseteq \mathbb{R}^d$ denote our finite data set where $\mathbf{x}_i$ is the d-dimensional feature vector representing the i-th sample. Also let $Y = \{y_i\}_{i=1}^{n+m}$ be their respective class labels where each y_i represents the numerical category varying between 1 to c, with c the number of classes. We distinguish indices of the labeled and unlabeled partitions of the data set by $\mathcal{L}$ and $\mathcal{U}$ (with $|\mathcal{L}| = n$ and $|\mathcal{U}| = m$), respectively, which are disjoint subsets of $\{1, ..., n+m\}$. Note that the *true* values of the labels $Y_\mathcal{L}$ are observed and denoted by $Y_\mathcal{L}^*$, hence $Y = Y_\mathcal{U} \cup Y_\mathcal{L}^*$. The initial classifier is trained based on these observed labels.

Labeling unlabeled samples is costly and time-consuming. Given a limited budget for this task, we wish to select $k \geq 1$ queries from $\mathcal{U}$ whose labeling leads to a new classifier with a significantly improved performance. Therefore we need an objective *set function* $f : 2^\mathcal{U} \to \mathbb{R}$ defined over subsets of the unlabeled indices $\mathcal{A} \subseteq \mathcal{U}$. The goal of batch active learning is then to choose a subset $\mathcal{A}$ with a given cardinality k that maximizes the objective given the current model that is trained based on $Y_\mathcal{L}^*$. This can be formulated as the following constrained combinatorial optimization:

$$\mathcal{A}^* = \underset{\mathcal{A}\subseteq\mathcal{U},\ |\mathcal{A}|=k}{\arg\max}\ f(\mathcal{A}). \tag{1}$$

We aim to choose the queries $\mathcal{A}$ whose labels $Y_\mathcal{A}$ give the highest amount of information about labels of the remaining samples $Y_{\mathcal{U}-\mathcal{A}}$. In other words, the goal is to maximize the mutual information (MI) between the random sets $Y_\mathcal{A}$ and $Y_{\mathcal{U}-\mathcal{A}}$ given the observed labels $Y_\mathcal{L}^*$ and the features X:

$$f_{MI}(\mathcal{A}) = MI(Y_\mathcal{A}, Y_{\mathcal{U}-\mathcal{A}}|X, Y_\mathcal{L}^*) = H(Y_\mathcal{A}|X, Y_\mathcal{L}^*) - H(Y_\mathcal{A}|X, Y_\mathcal{L}^*, Y_{\mathcal{U}-\mathcal{A}}). \tag{2}$$

Let us focus on the first right hand side term $f_H(\mathcal{A}) := H(Y_\mathcal{A}|X, Y_\mathcal{L}^*)$, which is the joint entropy function used in active learning methods [8]. Note that maximizing f_H is equivalent to minimizing $H(Y_{\mathcal{U}-\mathcal{A}}|X, Y_\mathcal{L}^*, Y_\mathcal{A})$ (the actual objective introduced in [8]), since $H(Y_\mathcal{U}|X, Y_\mathcal{L}^*) = H(Y_\mathcal{A}|X, Y_\mathcal{L}^*) + H(Y_{\mathcal{U}-\mathcal{A}}|X, Y_\mathcal{L}^*, Y_\mathcal{A})$ and $H(Y_\mathcal{U}|X, Y_\mathcal{L}^*)$ is a constant with respect to $\mathcal{A}$. This objective is expensive to calculate due to the complexity of computing the joint posterior $\mathbb{P}(Y_\mathcal{A}|X, Y_\mathcal{L}^*)$. However, using a point estimation, such as maximum likelihood, to train the classifier's parameter $\boldsymbol{\theta}$, one can say that $\boldsymbol{\theta}$ is deterministically equal to the maximum likelihood estimation (MLE) point estimate $\hat{\boldsymbol{\theta}}_n$ given X and $Y_\mathcal{L}^*$. Then we can rewrite the posterior as $\mathbb{P}(Y_\mathcal{A}|X, Y_\mathcal{L}^*) = \int \mathbb{P}(Y_\mathcal{A}|\boldsymbol{\theta}) \cdot \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_n) d\boldsymbol{\theta} = \mathbb{P}(Y_\mathcal{A}|\hat{\boldsymbol{\theta}}_n)$. Since in most discriminative classifiers the labels are assumed to be independent given the parameter, one can write $\mathbb{P}(Y_\mathcal{A}|\hat{\boldsymbol{\theta}}_n) = \prod_{i\in\mathcal{A}} \mathbb{P}(y_i|\hat{\boldsymbol{\theta}}_n)$. This simplifies the computation of the joint entropy to the sum of sample entropy contributions:

$$f_H(\mathcal{A}) = H(Y_\mathcal{A}|X, Y_\mathcal{L}^*) = H(Y_\mathcal{A}|\hat{\boldsymbol{\theta}}_n) = \sum_{j\in\mathcal{A}} H(y_j|\hat{\boldsymbol{\theta}}_n), \tag{3}$$

which is straightforward to compute having the pmf's $\mathbb{P}(y_i|\hat{\boldsymbol{\theta}}_n)$. Equation (3) implies that maximizing f_H can be separated into several individual maximizations, hence does not take into account the redundancy among the selected queries. Thus, in different related studies heuristics are added to cope with this issue. MI in Equation (2), on the other hand, removes this shortcoming by introducing a second term which conditions over the unobserved random variable $Y_{\mathcal{U}-\mathcal{A}}$, as well

as the observed $Y^*_{\mathcal{L}}$. This conditioning prevents the labels in $Y_{\mathcal{A}}$ from becoming independent, and therefore automatically incorporates the diversity among the queries (see next section for details of evaluating this term).

Unfortunately, maximizing f_{MI} for $k > 1$ is NP-hard (the optimization hurdle). Relaxing combinatorial optimizations into continuous spaces is a common technique to make the computations tractable [13], however these methods still involve a final discretization step that often includes using heuristics. In the following sections, we introduce our strategies to overcome the practical hurdles in MI-based active learning algorithms by introducing (1) pessimistic/optimistic approximations of MI; and (2) *submodular* maximization algorithms that allow us to perform the computations within the discrete domain.

2.2. Evaluating Mutual Information

In this Section, we address the hurdle of evaluating MI between non-singleton subset of labels. This objective, formulated in Equation (2), is also equal to

$$f_{MI}(\mathcal{A}) \;=\; H(Y_{\mathcal{U}-\mathcal{A}}|X, Y^*_{\mathcal{L}}) \;-\; H(Y_{\mathcal{U}-\mathcal{A}}|X, Y^*_{\mathcal{L}}, Y_{\mathcal{A}}), \tag{4}$$

due to MI's symmetry. We prefer this equation, since usually we have $|Y_{\mathcal{A}}| = k \ll |Y_{\mathcal{U}}|$ and thus it leads to a more computationally efficient problem. Note that the first term in the right hand side of Equation (4) can be evaluated similar to Equation (3). The major difficulty we need to handle in Equation (4) is the computation of the second term, which requires considering all possible label assignments to $Y_{\mathcal{A}}$. To make this computationally tractable, we propose to use a greedy strategy based on two variants: pessimistic and optimistic approximations of MI. To see this we focus on the second term:

$$H(Y_{\mathcal{U}-\mathcal{A}}|X, Y^*_{\mathcal{L}}, Y_{\mathcal{A}}) = \sum_{J\in\{1,\dots,c\}^{|\mathcal{A}|}} \mathbb{P}(Y_{\mathcal{A}} = J|X, Y^*_{\mathcal{L}}) \;\cdot\; H(Y_{\mathcal{U}-\mathcal{A}}|X, Y^*_{\mathcal{L}}, Y_{\mathcal{A}} = J) \tag{5}$$

where $\{1,\dots,c\}^{|\mathcal{A}|}$ is the set of all possible class label assignments to the samples in $\mathcal{A}$. For example, if $\mathcal{A}$ has three samples ($|\mathcal{A}| = 3$) and $c = 2$, then this set would be equal to $\{\{1,1,1\},\{2,1,1\},\{1,2,1\},\{2,2,1\},\{2,2,2\},\{1,2,2\},\{2,1,2\},\{1,1,2\}\}$. For each fixed label permutation J, the classifier should be retrained after adding the new labels $Y_{\mathcal{A}} = J$ to the training labels $Y^*_{\mathcal{L}}$ in order to compute the conditional entropy $H(Y_{\mathcal{U}-\mathcal{A}}|X, Y^*_{\mathcal{L}}, Y_{\mathcal{A}} = J)$. It is also evident from the example above that the number of possible assignments J to $Y_{\mathcal{A}}$ is c^k. Therefore, the number of necessary classifier updates grows exponentially with $|Y_{\mathcal{A}}| = k$. This is computationally very expensive and makes Equation (5) impractical. Alternatively, we can replace the expectation in Equation (5) with a minimization/maximization to get a *pessimistic/optimistic* approximation of MI. Such a replacement enables us to employ efficient greedy approaches to estimate f_{MI} in a conservative/aggressive manner. The greedy approach that we use here is compatible with the iterative nature of the optimization Algorithms 1 and 2 (described in Section 2.3). In the remainder of this Section, we focus on the pessimistic approximation. Similar equations can be derived for the optimistic case. The first step is replacing the weighted summation in Equation (5) by a maximization:

$$f^{pess}_{MI}(\mathcal{A}) \;=\; H(Y_{\mathcal{U}-\mathcal{A}}|X, Y^*_{\mathcal{L}}) - \max_{J\in\{1,\dots,c\}^{|\mathcal{A}|}} H(Y_{\mathcal{U}-\mathcal{A}}|X, Y^*_{\mathcal{L}}, Y_{\mathcal{A}} = J) \tag{6}$$

Note that $f^{pess}_{MI}(\mathcal{A})$ is always less than or equal to f_{MI}. Equation (6) still needs the computation of the conditional entropy for all possible assignments J. However, it enables us to use greedy approaches to approximate $f^{pess}_{MI}(\mathcal{A})$ for any candidate query set $\mathcal{A} \subseteq \mathcal{U}$, as described below.

Without loss of generality, suppose that $\mathcal{A}$, with size $|\mathcal{A}| = k$ ($1 \leq k \leq m$), can be shown element-wise as $\mathcal{A} = \{u_1,\dots,u_k\}$. Define $\mathcal{A}_t = \{u_1,\dots,u_t\}$ for any $t \leq k$ (hence $\mathcal{A}_k = \mathcal{A}$). In the

first iteration we can evaluate Equation (6) simply for the singleton $f_{MI}^{pess}(\{u_1\})$ and store $\hat{y}_{u_1}$, the assignment to y_{u_1} which maximizes the conditional entropy in Equation (6):

$$\begin{aligned} f_M^{pess}(\{u_1\}) &= H(Y_{\mathcal{U}-\{u_1\}}|X, Y_{\mathcal{L}}^*) - \max_{j\in\{1,\ldots,c\}} H(Y_{\mathcal{U}-\{u_1\}}|X, Y_{\mathcal{L}}^*, y_{u_1}=j) \\ &= H(Y_{\mathcal{U}-\{u_1\}}|X, \hat{\boldsymbol{\theta}}_n) - H(Y_{\mathcal{U}-\{u_1\}}|X, Y_{\mathcal{L}}^*, y_{u_1}=\hat{y}_{u_1}), \end{aligned} \tag{7}$$

where we used Equation (3) to substitute the first term with $H(Y_{\mathcal{U}-\{u_1\}}|\hat{\boldsymbol{\theta}}_n)$. Note that the second term in Equation (7) requires c times of retraining the classifier with the newly added class label $y_{u_1}=j$ for all possible $j\in\{1,\ldots,c\}$. In practice, the retraining process can be very time-consuming. Here, instead of retraining the classifier from scratch, we leverage the current estimate of the classifier's parameter vector and take one quasi-Newton step to update this estimate:

$$\tilde{\boldsymbol{\theta}}_{n+1} := \hat{\boldsymbol{\theta}}_n - \mathbf{H}_{n+1}^{-1}\cdot\mathbf{g}_{n+1}, \tag{8}$$

where $\mathbf{g}_{n+1}$ and $\mathbf{H}_{n+1}$ are the gradient vector and Hessian matrix of the log-likelihood function of our classifier given the labels $Y_{\mathcal{L}}^*\cup\{y_{u_1}=j\}$. Then we use the approximation

$$H(Y_{\mathcal{U}-\{u_1\}}|X, Y_{\mathcal{L}}^*, y_{u_1}=\hat{y}_{u_1}) = H(Y_{\mathcal{U}-\{u_1\}}|X, \boldsymbol{\theta}_{n+1}) \approx H(Y_{\mathcal{U}-\{u_1\}}|X, \tilde{\boldsymbol{\theta}}_{n+1}). \tag{9}$$

In Appendix, we derive the update equation in case a multinomial logistic regression is used as the discriminative classifier. Specifically, we will see that $\mathbf{g}_{n+1}$ and $\mathbf{H}_{n+1}^{-1}$ can be obtained efficiently from $\mathbf{g}_n$ and $\mathbf{H}_n^{-1}$.

If $k=1$, we are done. Otherwise, to move from iteration $t-1$ to t ($1<t\leq k$), $f_{MI}^{pess}(\mathcal{A}_{t-1}\cup\{u_t\})$ will be approximated from the previous iterations:

$$f_{MI}^{pess}(\mathcal{A}_t) \approx H\left(Y_{\mathcal{U}-\mathcal{A}_t}|\hat{\boldsymbol{\theta}}_n\right) - \max_{j\in\{1,\ldots,c\}} H\left(Y_{\mathcal{U}-\mathcal{A}_t}|X, Y_{\mathcal{L}}^*, \hat{Y}_{\mathcal{A}_{t-1}}, y_{u_t}=j\right), \tag{10}$$

where $\hat{Y}_{\mathcal{A}_{t-1}} = \{\hat{y}_{u_1},\ldots,\hat{y}_{u_{t-1}}\}$ are the assignments maximizing the conditional entropy that are stored from the previous iterations, such that the i-th element $\hat{y}_{u_i}$ is the assignment stored for $u_i=\mathcal{A}_i-\mathcal{A}_{i-1}(1\leq i\leq t)$. Note that Equation (10) is an approximation of the pessimistic MI, as is defined by Equation (6), however, in order to keep the notations simple we use the same notation f_{MI}^{pess} for both. Moreover, similar to Equation (7) there are c time of classifier updates involved in the computation of Equation (10). To complete iteration t, we make $\hat{y}_{u_t}$ equal to the assignment to y_{u_t} that maximizes the second term in Equation (10) and add it to $\hat{Y}_{\mathcal{A}_{t-1}}$ to form $\hat{Y}_{\mathcal{A}_t}$.

As in the first iteration, the conditional entropy term in Equation (10) is estimated by using the set of parameters obtained from the quasi-Newton step:

$$H\left(Y_{\mathcal{U}-\mathcal{A}_t}|X, Y_{\mathcal{L}}^*, \hat{Y}_{\mathcal{A}_{t-1}}, y_{u_t}=j\right) \approx H\left(Y_{\mathcal{U}-\mathcal{A}_t}|X, \tilde{\boldsymbol{\theta}}_{n+t}\right), \tag{11}$$

where

$$\tilde{\boldsymbol{\theta}}_{n+t} = \tilde{\boldsymbol{\theta}}_{n+t-1} - \mathbf{H}_{n+t}^{-1}\cdot\mathbf{g}_{n+t}. \tag{12}$$

Considering Equations (7) and (10) as the greedy steps of approximating f_{MI}, we see that the number of necessary classifier updates are $c\cdot k$, since there are k iterations each of which requires c times of retraining the classifier. Thus, the computational complexity reduced from the exponential cost in the exact formulation Equation (5) to the linear cost in the greedy approximation.

Similar to Equation (10), for the optimistic approximation, we will have:

$$f_{MI}^{opt}(\mathcal{A}_t) = H\left(Y_{\mathcal{U}-\mathcal{A}_t}|\hat{\boldsymbol{\theta}}_n\right) - \min_{j\in\{1,\ldots,c\}} H\left(Y_{\mathcal{U}-\mathcal{A}_t}|X, Y_{\mathcal{L}}^*, \hat{Y}_{\mathcal{A}_{t-1}}, y_{u_t}=j\right), \tag{13}$$

where $\hat{Y}_{\mathcal{A}_{t-1}} = \{\hat{y}_{u_1}, ..., \hat{y}_{u_{t-1}}\}$ is the set of class assignments minimizing the conditional entropy that are stored from the previous iterations. Clearly, the reduction of the computational complexity remains the same in the optimistic formulation.

Let us emphasize that, from the definitions of f_{MI}^{pess} and f_{MI}^{opt}, we always have the following inequality

$$f_{MI}^{pess}(\mathcal{A}) \leq f_{MI}(\mathcal{A}) \leq f_{MI}^{opt}(\mathcal{A}) \quad ,\forall \mathcal{A} \subseteq \mathcal{U}. \tag{14}$$

The first (or second) inequality turns to equality, if the results of averaging in conditional entropy in Equation (5) is equal to maximization (or minimization) involved in the approximations. This is equivalent to saying that the posterior probability $\mathbb{P}(Y_{\mathcal{A}}|X, Y_{\mathcal{L}}^*)$ is a degenerative distribution concentrated at the assignment $Y_{\mathcal{A}} = J$ that maximizes (or minimizes) the conditional entropy. Furthermore, if the posterior is a uniform distribution, giving the same posterior probability to all possible assignments $J \in \{1, ..., c\}^{|\mathcal{A}|}$, then the averaging, minimization and maximization lead to the same numerical result and therefore we get $f_{MI}^{pess} = f_{MI}^{opt} = f_{MI}$.

In theory, the value of MI between any two random variables is non-negative. However, because of the approximations made in computing the pessimistic or optimistic evaluations of MI, it is possible to get negative values depending on the distribution of the data. Therefore, after going through all the elements of $\mathcal{A}$ in evaluating f_{MI}^{pess} (or f_{MI}^{opt}), we take the maximum between the approximations of $f_{MI}^{pess}(\mathcal{A})$ (or $f_{MI}^{opt}(\mathcal{A})$) and zero to ensure its non-negativity.

2.3. *Randomized* vs. *Deterministic Submodular Optimizations*

In this section, we begin by reviewing the basic definitions regarding submodular set functions, and see that both f_{MI} and f_H satisfy submodularity condition. We then present two methods for submodular maximization: a deterministic and a randomized approach. The latter is applicable to submodular and monotone set functions such as f_H. But f_{MI} is not monotone in general, hence we present the randomized approach for this objective.

Definition 1. A set function $f : 2^{\mathcal{U}} \to \mathbb{R}$ is said to be *submodular* if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq \mathcal{U}. \tag{15}$$

We call f *supermodular* if the inequality in Equation (15) is reversed. In many occasions, it is easier to use an equivalent definition, which uses the notion of *discrete derivative* defined as:

$$\rho_f(\mathcal{A}, u) := f(\mathcal{A} \cup \{u\}) - f(\mathcal{A}) \quad ,\forall \mathcal{A} \subseteq \mathcal{U}, \ u \in \mathcal{U}. \tag{16}$$

Proposition 2. *Let $f : 2^{\mathcal{U}} \to \mathbb{R}$ be a set function. f is submodular if and only if we have*

$$\rho_f(\mathcal{A}, u) \geq \rho_f(\mathcal{B}, u), \quad \forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}, \ u \in \mathcal{U} - \mathcal{B}. \tag{17}$$

This equips us to show the submodularity of joint entropy and MI:

Theorem 3. *The set functions f_H and f_{MI}, defined in Equations (3) and (2) above, are submodular.*

Proof. It is straightforward to check the submodularity of f_H and therefore the first term of MI formulation in Equation (2). It remains to show that $g(\mathcal{A}) := H(Y_{\mathcal{A}}|X, Y_{\mathcal{L}}^*, Y_{\mathcal{U}-\mathcal{A}})$, the second term with the opposite sign, is supermodular. Let us first write the discrete derivative of the function g:

$$\begin{aligned} \rho_g(\mathcal{A}, u) &= g(\mathcal{A} \cup \{u\}) - g(\mathcal{A}) \\ &= H(Y_{\mathcal{A}\cup\{u\}}|X, Y_{\mathcal{L}}^*, Y_{\mathcal{U}-\mathcal{A}\cup\{u\}}) - H(Y_{\mathcal{A}}|X, Y_{\mathcal{L}}^*, Y_{\mathcal{U}-\mathcal{A}}) \\ &= H(y_u|X, Y_{\mathcal{L}}^*, Y_{\mathcal{U}-\mathcal{A}\cup\{u\}}), \end{aligned} \tag{18}$$

which holds for any $u \notin \mathcal{A} \subseteq \mathcal{U}$. Here, we used that the joint entropy of two sets of random variables A and B can be written as $H(A,B) = H(A) + H(B|A)$. Now take any superset $\mathcal{B} \supseteq \mathcal{A}$, which does not contain $u \in \mathcal{U}$. From $\mathcal{B} \supseteq \mathcal{A}$, we have $Y_{\mathcal{U}-\mathcal{A}\cup\{u\}} \subseteq Y_{\mathcal{U}-\mathcal{B}\cup\{u\}}$ and therefore $\rho_g(\mathcal{A},u) - \rho_g(\mathcal{B},u) = H(y_u|X, Y^*_{\mathcal{L}}, Y_{\mathcal{U}-\mathcal{A}\cup\{u\}}) - H(y_u|X, Y^*_{\mathcal{L}}, Y_{\mathcal{U}-\mathcal{B}\cup\{u\}}) \leq 0$ implying supermodularity of g. □

Although submodular functions can be minimized efficiently, they are NP-hard to maximize [19], and therefore we have to use approximate algorithms. Next, we briefly discuss the classical approximate submodular maximization method widely used in batch querying [9,11,12,16,17]. This greedy approach, we call *deterministic* throughout this paper, is first proposed in the seminal work of [20] (shown in Algorithm 1) and its performance is analyzed for *monotone* set functions as follows:

Definition 4. The set function $f : 2^{\mathcal{U}} \to \mathbb{R}$ is said to be *monotone (nondecreasing)* if for every $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$ we have $f(\mathcal{A}) \leq f(\mathcal{B})$.

Theorem 5. *Let $f : 2^{\mathcal{U}} \to \mathbb{R}$ be a submodular and nondecreasing set function with $f(\varnothing) = 0$, $\mathcal{A}$ be the output of Algorithm 1 and $\mathcal{A}^*$ be the optimal solution to the problem in Equation (1). Then we have:*

$$f(\mathcal{A}) \;\geq\; \left[1 - \left(\frac{k-1}{k}\right)^k\right] f(\mathcal{A}^*) \;\geq\; \left(1 - \frac{1}{e}\right) f(\mathcal{A}^*). \tag{19}$$

Algorithm 1: The deterministic approach

Inputs: The objective function f, the unlabeled indices $\mathcal{U}$, the query batch size $k > 0$
Outputs: a subset of unlabeled indices $\mathcal{A} \subseteq \mathcal{U}$ of size k

```
   /* Initializations                                                    */
1  A_0 ← ∅
2  U_0 ← U
   /* Starting the Iterations                                            */
3  for t = 1 → k do
       /* Local maximization                                            */
4      u_t ← arg max_{u∈U_{t-1}} f(A_{t-1} ∪ {u})
       /* Updating the Loop Variables                                   */
5      A_t ← A_{t-1} ∪ {u_t}
6      U_t ← U − A_t
7  return A = A_k
```

The proof is given by [20] and [21]. Among the assumptions, $f(\varnothing) = 0$ can always be assumed since maximizing a general set function $f(\mathcal{A})$ is equivalent to maximizing its adjusted version $g(\mathcal{A}) := f(\mathcal{A}) - f(\varnothing)$ which satisfies $g(\varnothing) = 0$. Nemhauser *et al.* [22] also showed that Algorithm 1 gives the *optimal* approximate solution to the problem in (1) for nondecreasing functions such as f_H. However, f_{MI} is not monotone in general and therefore Theorem 5 is not applicable. To imagine non-monotonicity of f_{MI}, it suffices to imagine that $f_{MI}(\varnothing) = f_{MI}(\mathcal{U}) = 0$.

Recently several algorithms have been proposed for approximate maximization of *nonnegative* submodular set functions, which are not necessarily monotone. Feige *et al.* [23] made the first attempt towards this goal by proposing a $(2/5)$-approximation algorithm and also proving that $1/2$ is the optimal approximation factor in this case. Buchbinder *et al.* [24] could achieve this optimal bound in expectation by proposing a randomized iterative algorithm. However, these algorithms are designed for *unconstrained* maximization problems. Later, Buchbinder *et al.* [25] devised a $(1/e)$-approximation randomized algorithm with cardinality constraint, which is more suitable for batch active learning.

A pseudocode of this approach is shown in Algorithm 2 where instead of selecting the sample with maximum objective value at each iteration, the best k samples are identified (line 4) and one of them is chosen randomly (line 5). Such a randomized procedure provides a $(1/e)$-approximation algorithm for maximizing a nonnegative submodular set function such as f_{MI}:

Theorem 6. *Let $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}$ be a submodular nonnegative set function and $\mathcal{A}$ be the output of Algorithm 2. Then if $\mathcal{A}^*$ is the optimal solution to the problem in (1) we have:*

$$\mathbb{E}[f(\mathcal{A})] \;\geq\; \left(1-\frac{1}{k}\right)^{k-1} f(\mathcal{A}^*) \;\geq\; \frac{1}{e}\; f(\mathcal{A}^*). \tag{20}$$

The proof can be found in [25] and our supplementary document. In order to be able to select k samples from $\mathcal{U}_t$ to form $\mathcal{M}_t$ for all t, it suffices to ensure that the smallest unlabeled set that we sample from $\mathcal{U}_{k-1}$ has enough members, *i.e.*, $k \leq |\mathcal{U}_{k-1}| = |\mathcal{U}| - k + 1$ hence $k \leq (|\mathcal{U}|+1)/2$.

Observe that although the assumptions in Theorem 6 are weaker than those in Theorem 5, the bound shown in Equation (20) is also looser than that in Equation (19). However, interestingly, it is proven that inequality Equation (19) will still hold for Algorithm 2 if the monotonicity of f is satisfied (see the Theorem 3.1. in [25]). Thus, the randomized Algorithm 2 is expected to be performing similar to Algorithm 1 for monotone functions.

Algorithm 2: The randomized approach

Inputs: The objective function f, the unlabeled indices $\mathcal{U}$, the query batch size $k > 0$
Outputs: a subset of unlabeled indices $\mathcal{A} \subseteq \mathcal{U}$ of size k

```
   /* Initializations                                                    */
1  A_0 ← ∅
2  U_0 ← U
   /* Starting the Iterations                                            */
3  for t = 1 → k do
       /* Selecting k points with highest f values                       */
4      M_t ← arg max_{M⊆U_{t-1}, |M|=k}  Σ_{u∈M} f(A_{t-1} ∪ {u})
       /* Random Selection                                               */
5      u_t ← RANDOM(M_t)
       /* Updating the Loop Variables                                    */
6      A_t ← A_{t-1} ∪ {u_t}
7      U_t ← U − A_t
8  return A = A_k
```

Algorithms 1 and 2 are equivalent for sequential querying ($k = 1$). Also note that in both algorithms, the variables u_t in iteration t, is determined by deterministic or stochastic maximization of $f(\mathcal{A}_{t-1} \cup \{u\})$. Fortunately, such maximization needs only computations in the form of Equation (10) or Equation (13) when $f = f_{MI}^{pess}$ or f_{MI}^{opt}. These computations can be done easily provided that the gradient vector $\mathbf{g}_{n+t-1}$ and inverse-Hessian matrix $\mathbf{H}_{n+t-1}^{-1}$ have been stored from the previously selected subset $\mathcal{A}_{t-1}$. The updated gradient and inverse-Hessian that are used to compute $f(\mathcal{A}_{t-1} \cup \{u\})$ are different for each specific $u \in \mathcal{U}_{t-1}$. We only save those associated with the local maximizer, that is u_t, as $\mathbf{g}_{n+t}$ and $\mathbf{H}_{n+t}^{-1}$ to be used in the next iteration.

2.4. Total Complexity Reduction

We measure the complexity of a given querying algorithm in terms of the required number of classifier updates. This makes our analysis general and independent of the updating procedure, which can be done in several possible ways. As we discussed in the last section, we chose to perform a single step of quasi Newton in Equation (8) but alternatively one can use full training or any other numerical parameter update.

Consider the following optimization problems:

$$\underset{\substack{\mathcal{A}\subseteq\mathcal{U}\\|\mathcal{A}|=k}}{\arg\max} \quad f_{MI}(\mathcal{A}), \tag{21a}$$

$$\text{greedy} \quad \underset{\substack{\mathcal{A}\subseteq\mathcal{U}\\|\mathcal{A}|=k}}{\arg\max} \quad \tilde{f}_{MI}(\mathcal{A}), \tag{21b}$$

where "greedy *arg max*" denotes the greedy maximization operator that uses Algorithm 1 or 2 to maximize the objective, and $\tilde{f}_{MI}$ is either f_{MI}^{pess} or f_{MI}^{opt}. Note that Equation (21a) formulates the global maximization of the exact MI function and Equation (21b) shows the optimization in our framework, that is a greedy maximization of the pessimistic/optimistic MI approximations. In the following remark, we compare the complexity of solving the two optimizations in Equation (21) in terms of the number of classifier updates required for obtaining the solutions.

Remark 1. For a fixed k, the number of necessary classifier updates for solving Equation (21a) increases with order k, whereas for Equation (21b) it changes linearly.

Proof. As is explained in Section 2.2, the number of classifier updates for computing $f_{MI}(\mathcal{A})$ without any approximations, is c^k. Moreover, in order to find the global maximizer of MI, f_{MI} needs to be evaluated at all subsets of $\mathcal{U}$ with size k. There are $\binom{m}{k} = O\left(m^k\right)$ of such subsets (recall that $m = |\mathcal{U}|$). Hence, the total number of classifier update required for global maximization f_{MI} is of order $O\left((m \cdot c)^k\right)$.

Now, regarding Equation (21b), recall from Section 2.2 that if $\mathbf{g}_{n+t-1}$ and $\mathbf{H}^{-1}_{n+t-1}$ are stored from the previous iteration, computing $\tilde{f}_{MI}(\mathcal{A}_{t-1} \cup \{u_t\})$ needs only c classifier updates. However, despite the evaluation problem in Section 2.2, in computing line (4) of Algorithms 1 and 2, the next sample to add, that is u_t, is not given. In order to obtain u_t, $\tilde{f}_{MI}$ is to be evaluated at all the remaining samples in $\mathcal{U}_{t-1}$. Since, $|\mathcal{U}_{t-1}| = m - t + 1$, the number of necessary classifier updates in the t-th iteration is $c \cdot (m - t + 1)$. Both algorithms run k iterations that results the following total number of classifier updates:

$$cm + c(m-1) + ... + c(m-k+1) \;=\; ck\left(m - \frac{k+1}{2}\right) \;=\; O(ckm).$$

□

2.5. Further Speed-Up

Here, we show that the total complexity of our proposed MI-based querying approaches is linear, in contrast with the exponential cost of the exact formulation which is not practical.

Even after approximating f_{MI} using the pessimistic or optimistic formulations, MI-based algorithms can be significantly slow for large data sets. In order to further scale up our algorithm, induced by [18], we first selects a subset of the unlabeled samples by only choosing the most β

uncertain samples (where $\beta \in \mathbb{Z}^+$). We then ran our MI-based algorithm over such filtered data. More formally, the input set of unlabeled indices to Algorithms 1 and 2 will be

$$\mathcal{U}_f = \underset{\substack{U \subseteq \mathcal{U} \\ |U|=\beta}}{\arg\max} \sum_{u \in U} f_H(\{u\}). \tag{22}$$

It is evident that for $\beta = |\mathcal{U}|$ the filtered data will be equal to the original unlabeled pool $\mathcal{U}_f = \mathcal{U}$. From now on, we add the adjective *filtered* to any querying algorithm that is preceded by reduction of the unlabeled data into the samples with high uncertainty as described above.

3. Results and Discussion

In this section, we show our experimental results over several data sets on three different fields: medicine Section 3.1, image processing Section 3.2 and music harmony analysis Section 3.3. We ran our MI-based querying algorithms against entropy-based and random active learning benchmarks. In the following section, first we describe the data sets that have been used, then we explaining the experimental settings and present the numerical results.

3.1. Cardiotocography

This data set is downloaded from UCI repository [26] and contains 2126 fetal cardiotocograms each of which is represented by a 21-dimensional feature vector. The data is categorized into three classes based on the fetal states: normal, suspect and pathological (therefore $c = 3$). All the data samples are first projected into a 15-dimensional principal component analysis (PCA) subspace obtained from the unlabeled pool. The initial labeled data set chosen randomly in the beginning of each experiment, consists of 75 samples (25 samples per class). We will refer to this data set as "Cardio" in the following sections.

3.2. MNIST (Mixed National Institute of Standards and Technology)

This is an image database of handwritten digits 1 to 9 [27]. Here, we only use images for digits 1 to 4, hence $c = 4$. The data set, consisting of 20×20 images, is already divided into a testing/training partitions. In our experiments, these partitions are fixed as given, but each time the initial labeled data sets are randomly chosen from the training partition. The raw 400-dimensional feature vectors are projected into 10-dimensional PCA subspace constructed based on the training partition. After choosing only images of digits from 1 to 4, the size of the testing and training partitions are 4130 and 4159, respectively. We also set the size of our initial labeled data set $\mathcal{L}_0$ to 200 (50 samples per class).

3.3. Bach Choral Harmony

The other data set that we used for evaluating performance of the algorithms contains pitch information of time events of 60 chorales by Johann Sebastian Bach [28]. Each event is represented by pitch-wise and meter information and is assigned a chord class label. We selected the events associated with the five most frequent chords in the data set: D-major, G-major, C-major, F-major and A-major (hence $c = 5$); resulting a set of 2221 samples. Discarding pitch class of the bass notes and the metric information, we used the binary indicators of the pitch classes corresponding to equal-tempered 12 notes of the chromatic scale. This leads to a set of 12-dimensional binary feature vectors that are projected into 8-dimensional PCA subspace obtained based on the training data at each experiment. We will refer to this data set simply as "Bach" in the remaining sections.

3.4. Experimental Settings

From the previous sections, we have two methods of evaluating MI and two optimization techniques, leading to four different ways of doing MI-based querying, in all of which we used the

filtered pool of unlabeled samples $\mathcal{U}_f$ that is obtained with $\beta = 100$. Throughout this section, we distinguish different approaches involved in our experimental settings using the labels listed below:

- Pess-MI-Det: Pessimistic MI f_{MI}^{pess} with deterministic optimization (Algorithm 1);
- Pess-MI-Rand: Pessimistic MI f_{MI}^{pess} with randomized optimization (Algorithm 2);
- Opt-MI-Det: Optimistic MI f_{MI}^{opt} with deterministic optimization (Algorithm 1);
- Opt-MI-Rand: Optimistic MI f_{MI}^{opt} with randomized optimization (Algorithm 2);
- entropy: Entropy objective f_H with deterministic optimization (Algorithm 1);
- random: Random querying.

In sequential querying, where deterministic and randomized optimization algorithms are equivalent, we use Pess-MI and Opt-MI to refer to the MI-based objectives without mentioning the optimization type.

In running the querying experiments over all the data sets, we used a linear logistic regression as the core classifier. In case that the data under consideration is not already divided into testing/training partitions, we randomly generate such partitions in each experiment with fixed ratio of 3/7 (testing size to training size). The initial training data set $\mathcal{L}_0$ is randomly selected from the training partition and the rest of the training samples are considered as the unlabeled pool $\mathcal{U}$ from which the queries are to be selected in each querying iteration. Moreover, in each experiment we first reduce the dimensionality of the data using PCA over the unlabeled pool.

In the experiments, we iteratively select the query batches of either sizes $k = 1, 5, 10$ or 20, add the selected queries together with their class labels to the labeled data set and re-calculate the parameters of the classifier, which in turn, leads to an updated accuracy value based on the testing partition. For each value of k, we repeated running the experiments for 25 times, each time with a different random selection of testing/training partitions and a different initial labeled set $\mathcal{L}_0$. Hence, in total, we get 25 accuracy curves for each value of k. Ideally, we want an active learning algorithm whose accuracy curve increases as fast as possible, *i.e.*, obtaining a more accurate classifier with labeling fewer number of query batches.

In order to present the performance of the listed algorithms, we calculate the average and standard deviation (STD) of the 25 accuracy curves for each algorithm. Furthermore, for pairwise comparison between the MI-based approaches and the benchmarks for a fixed value of k, we perform two-sample one-tail T-tests over the accuracy values of the competing algorithms. Note that such hypothesis test can and should be done over the accuracy values calculated after each querying iteration t separately. Here, the assumption is that the accuracy levels generated from the 25 querying experiments at iteration t are independent from each other. Let η_t denote the random variable presenting the accuracy of the updated classifier after t times of running an MI-based querying algorithm and η'_t be a similar random variable for a competing non-MI-based method. Then, we consider the null and alternative hypotheses to be:

$$\begin{aligned} H_0: \quad & \mu(\eta_t) \leq \mu(\eta'_t) \\ H_1: \quad & \mu(\eta_t) > \mu(\eta'_t) \end{aligned} \tag{23}$$

where $\mu(\eta_t)$ and $\mu(\eta'_t)$ are the mean of the random variables η_t and η'_t. We perform such T-test for comparing all modes of MI-based objectives (see the list above) versus the entropy-based and random querying algorithms. Rejecting the null hypothesis implies that the new accuracy in the t-th iteration of an MI-based querying is not less than or equal to the case when we use a querying objective other than the approximating variants of f_{MI}. In other words, obtaining a smaller p-value for a T-test described above, means that with a higher probability the accuracy of the updated classifier is larger when using the MI-based approach for querying.

3.5. Numerical Results

The numerical results of running sequential active learning with different querying algorithms are shown in Figure 1 and the results of batch active learning with different batch sizes are shown in Figures 2 (for $k = 5$), 3 (for $k = 10$) and 4 (for $k = 20$). The figures show the average accuracy curves (first row of each figure), the standard deviation of the accuracy curves (second rows), and the resulting p-values of the hypothesis tests for comparison between the MI-based approaches and entropy-based (third rows) or random querying (fourth rows).

As it is mentioned before, the deterministic and randomized optimization algorithms described in Section 2.3 are equivalent in sequential querying. Therefore, we have only two variants of MI-based querying in Figure 1. This figure shows that for two data sets (MNIST and Bach) the MI-based approaches perform similar or sometimes even worse than the entropy-based benchmark. This can be explained by noting that the main shortcoming of using the entropy objective f_h, that is redundancy among the queries, is meaningful only when we have multiple samples in the batch, that is $k > 1$. However, they mostly outperform random querying. Another observation is that using optimistic approximation of MI gave better results both in terms of average accuracy and the hypotheses p-values. Recall that the optimistic approximation of MI tries to minimize the entropy over the class labels of the remaining samples in each iteration, while the pessimistic approximation uses maximization of this entropy. Hence, our conjecture for this observation is that the optimistic approach does more aggressive exploitation in comparison with the pessimistic variant, in the sense of choosing the queries from the set of samples that lead to a lower classifier entropy.

For batch mode active learning, the MI-based approaches generally show better performance. That is, their accuracy curves grow more rapidly than the benchmarks. When comparing against the entropy-based approach, for data sets Cardio and Bach, we observe that the p-values are small in the beginning iterations. Whereas for MNIST data set, low p-values are mostly seen in the middle or late iterations. Hence, the probability that MI-based variants outperform the entropy is high in early querying iterations, before the labeled training set becomes large, for the two former data sets, and in later iterations for MNIST. This behavior can also be seen from the plots of the average accuracy.

Regarding the comparison against the random benchmark, we see from the p-value plots more conspicuously that MI-based approaches generally outperform random with high confidence soon after the early iterations. However, the plots for Bach, show that this confidence decrease in late iterations, which is mainly due to the growth of the accuracies of random to the same level as the MI-based curves.

Whereas in sequential active learning the optimistic approach did a better job in comparison with the pessimistic variant, the difference between these two variants shrinks as the size of query batch increases (and so does the number of required approximation iterations). Our conjecture is that accumulation of the approximation error makes the performance of optimistic and pessimistic MI-based querying methods closer to each other, though still better than the benchmarks. Additionally, there is no significant difference between using deterministic or randomized optimization algorithms, which might be because of local monotonicity of the approximations f_{MI}. Also note that although there are large distinctions between MI-based variants when their p-values are large, we ignore those parts as uninformative regions, since they just imply that the probability of MI-based approaches outperforming the benchmarks is not high.

$k = 1$:

Cardio	MNIST	Bach

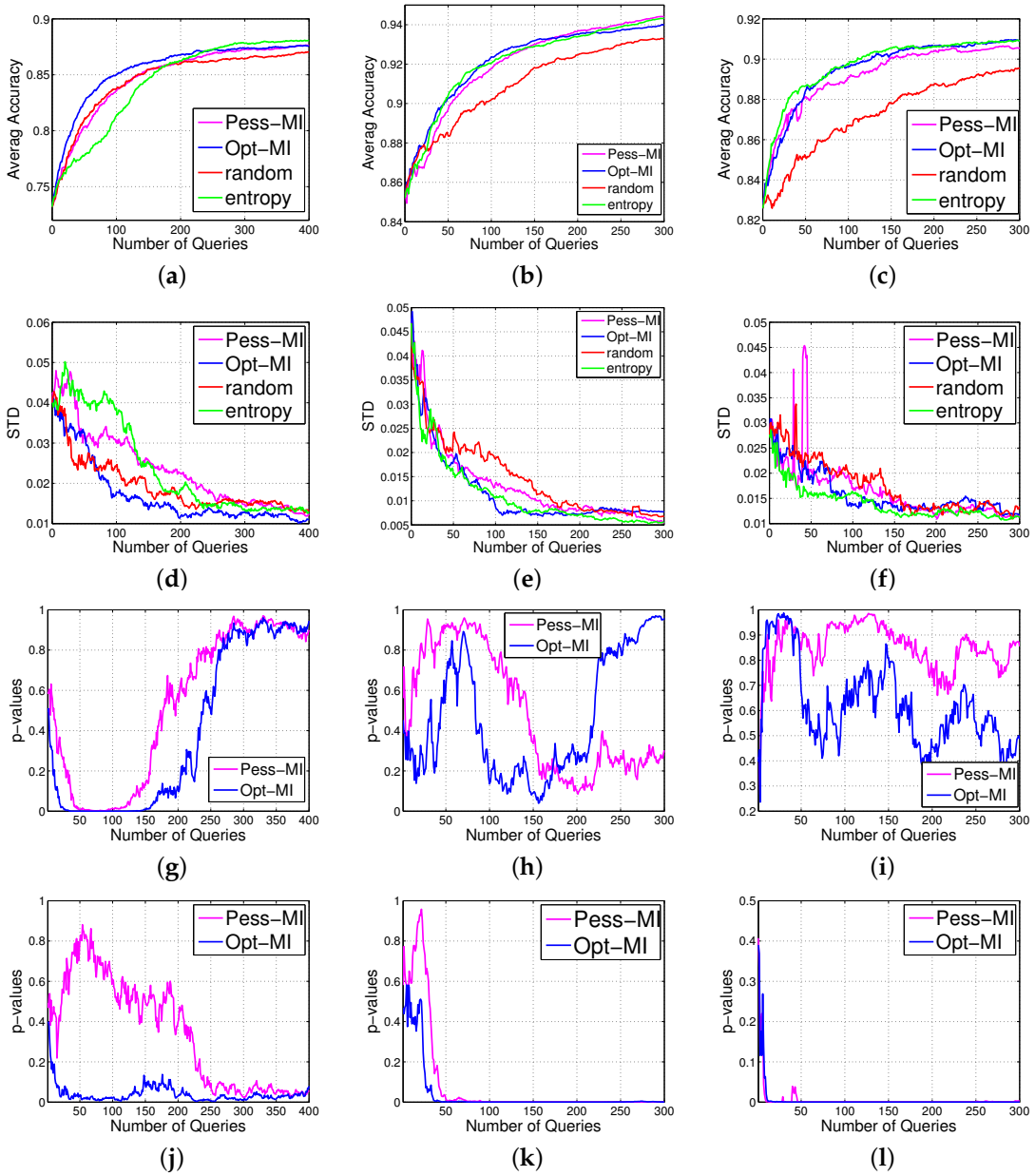

Figure 1. The experimental results of different querying approaches for sequential active learning ($k = 1$). (**a**) Average; (**b**) Average; (**c**) Average; (**d**) STD; (**e**) STD; (**f**) STD; (**g**) p-value: MI *vs.* Entropy; (**h**) p-value: MI *vs.* Entropy; (**i**) p-value: MI *vs.* Entropy; (**j**) p-value: MI *vs.* Random; (**k**) p-value: MI *vs.* Random; (**l**) p-value: MI *vs.* Random.

$k = 5$:

Cardio	MNIST	Bach

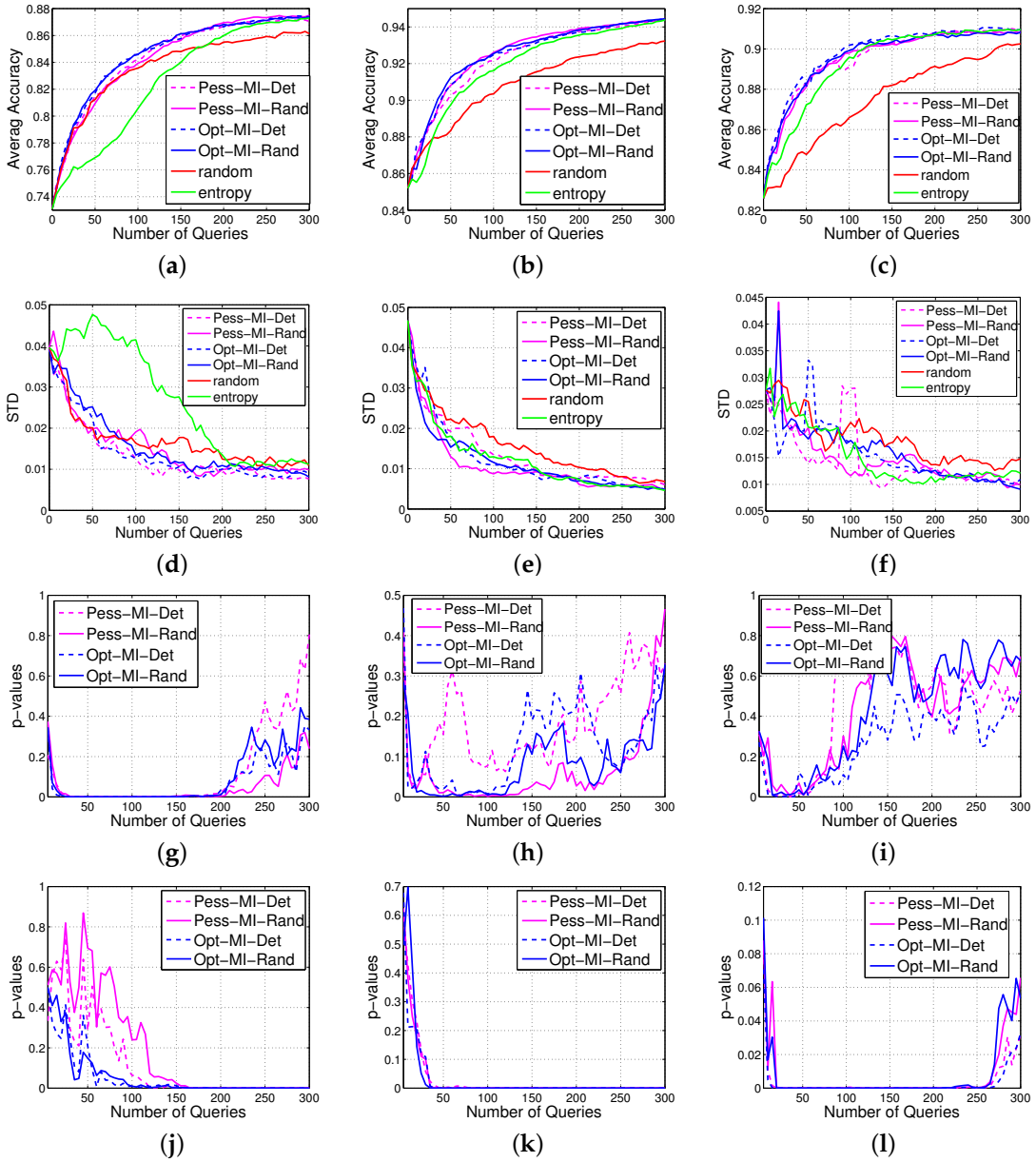

Figure 2. The experimental results of different querying approaches for batch active learning ($k = 5$). (**a**) Average; (**b**) Average; (**c**) Average; (**d**) STD; (**e**) STD; (**f**) STD; (**g**) p-value: MI *vs*. Entropy; (**h**) p-value: MI *vs*. Entropy; (**i**) p-value: MI *vs*. Entropy; (**j**) p-value: MI *vs*. Random; (**k**) p-value: MI *vs*. Random; (**l**) p-value: MI *vs*. Random.

$k = 10$:

Cardio	MNIST	Bach

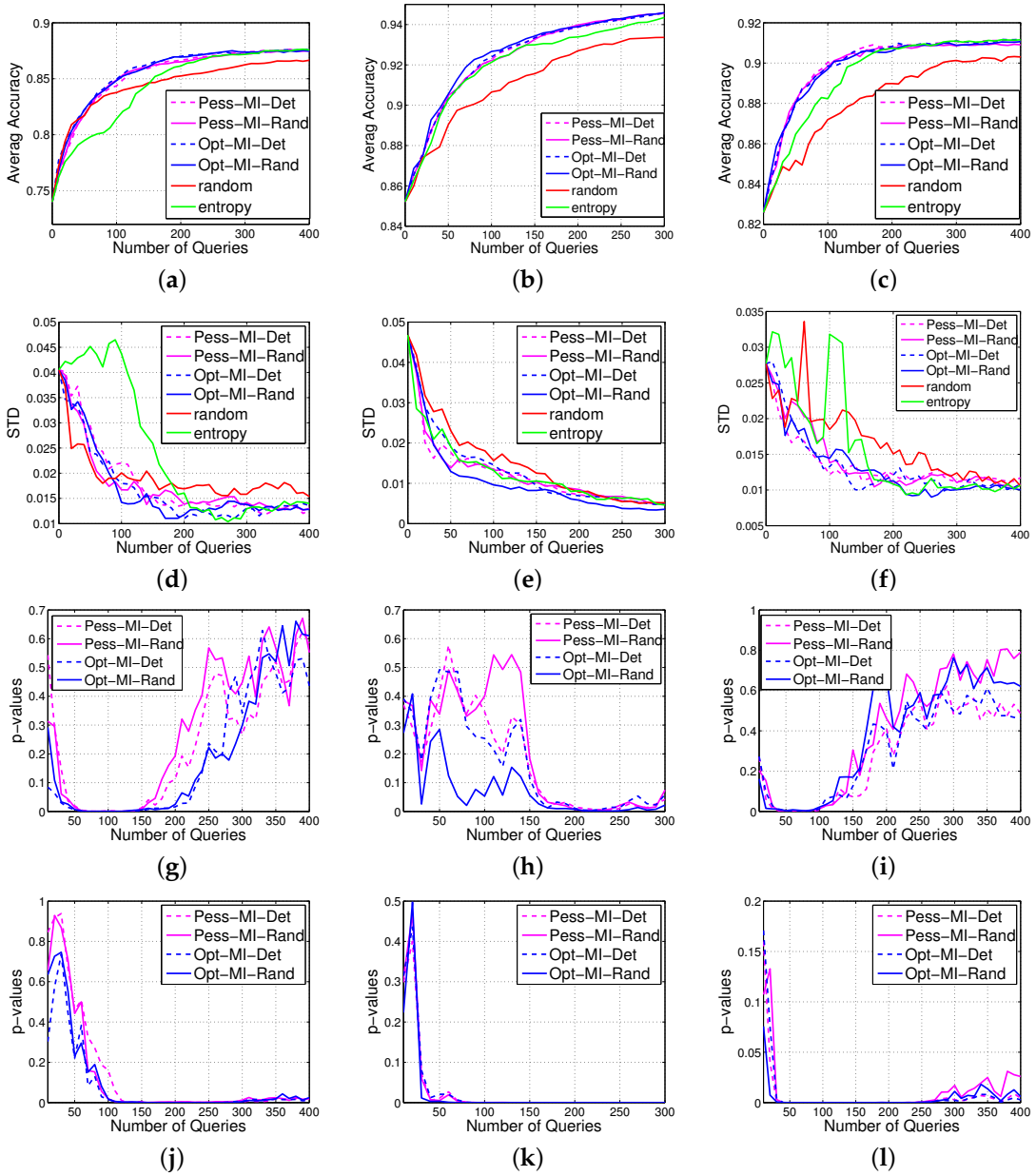

Figure 3. The experimental results of different querying approaches for batch active learning ($k = 10$). (**a**) Average; (**b**) Average; (**c**) Average; (**d**) STD; (e) STD; (**f**) STD; (g) p-value: MI *vs*. Entropy; (**h**) p-value: MI *vs*. Entropy; (**i**) p-value: MI *vs*. Entropy; (**j**) p-value: MI *vs*. Random; (**k**) p-value: MI *vs*. Random; (**l**) p-value: MI *vs*. Random.

$k = 20$:

Cardio	MNIST	Bach

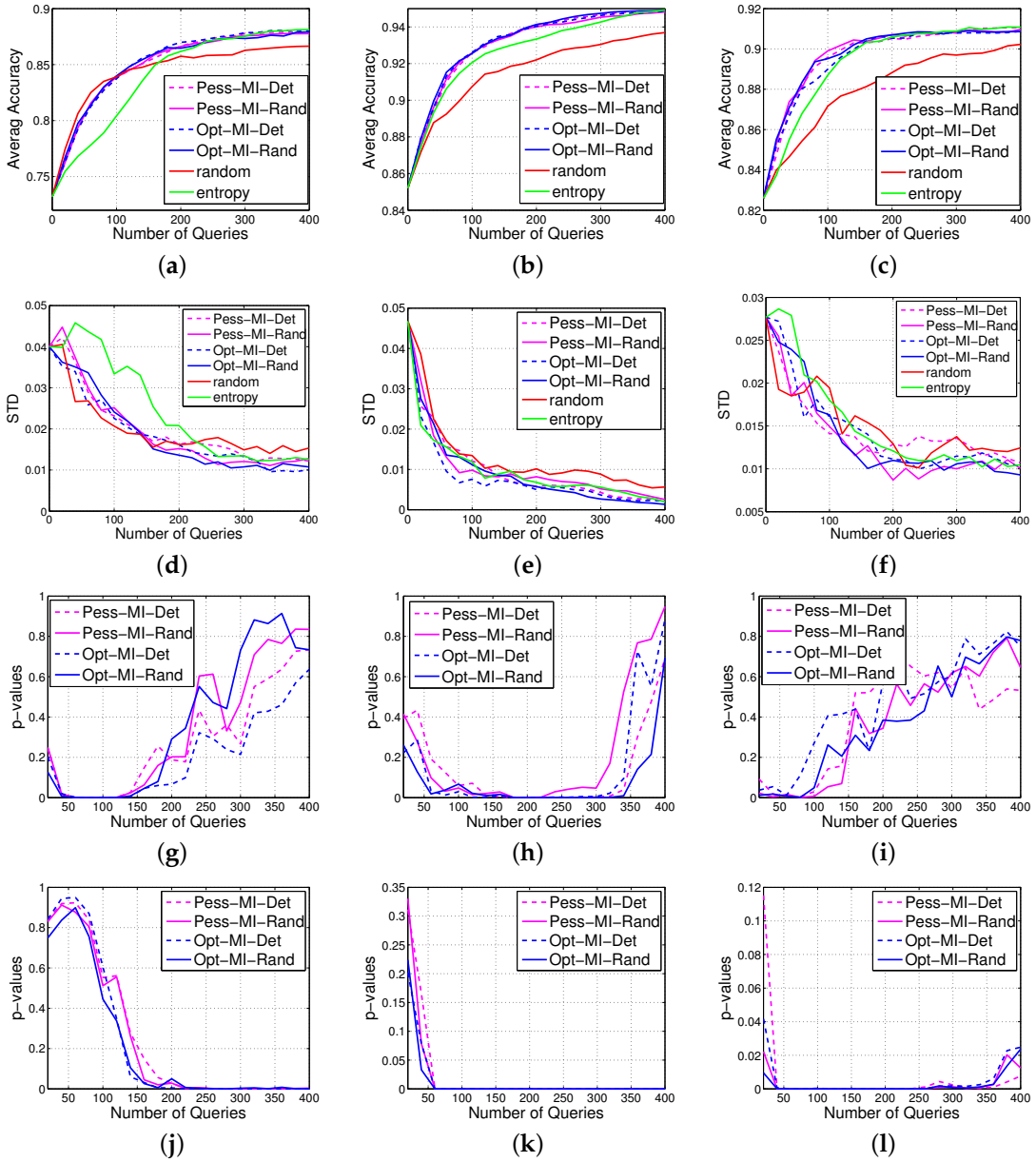

Figure 4. The experimental results of different querying approaches for batch active learning ($k = 20$). (**a**) Average; (**b**) Average; (**c**) Average; (**d**) STD; (**e**) STD; (**f**) STD; (**g**) p-value: MI *vs.* Entropy; (**h**) p-value: MI *vs.* Entropy; (**i**) p-value: MI *vs.* Entropy; (**j**) p-value: MI *vs.* Random; (**k**) p-value: MI *vs.* Random; (**l**) p-value: MI *vs.* Random.

4. Conclusions

Active learning based on reducing the model uncertainty is very popular, however most of the relevant objectives such as entropy and sample margin, do not take into account the diversity between the queries in case of batch active learning. Working with probabilistic classifiers, one natural replacement for entropy is mutual information (MI) between the class labels of the candidate queries and the remaining unlabeled samples. But hurdles in evaluating and efficient optimization of this

objective has failed its popularity in the learning community. In this paper we presented a framework for efficient querying based on this objective.

In our framework, we proposed pessimistic and optimistic approximations of MI by replacing the averaging operator inside the conditional entropy with maximizing and minimizing operators, respectively. This enabled us to efficiently estimate these values in a greedy fashion. Furthermore, in a consistent flow with the greedy estimation of MI, the optimization is also done in an iterative scheme. The iterative nature of the optimization decreased the computational complexity from $O\left((m \cdot c)^k\right)$, when no approximation is applied, to $O(ckm)$. Two different modes of optimization are suggested based on the existing algorithms in submodular maximization literature: one the classical deterministic greedy approach that has been already used in learning literature, and the other one a stochastic variant of it that is especially useful when monotonicity is absent in the submodular objective.

We generated experimental results using various real-world data sets in order to evaluate the performance of our MI-based querying approaches against the entropy-based and random querying benchmarks in terms of the accuracy growth rate during the querying iterations. These results show that MI-based algorithms outperformed the rest in most of the cases, especially when $k > 1$ (batch active learning). Furthermore, the optimistic variant outperformed the pessimistic one, especially for smaller batch sizes. As we increase the number of the queries and therefore the number of approximating iterations, this difference shrinks.

Acknowledgments: This work is primarily supported by NSF IIS-1118061. In addition, Todd Leen was supported under NSF 1258633, 1355603, and 1454956; Deniz Erdogmus was supported by NSF IIS-1149570, CNS-1136027 and SMA-0835976; and Jennifer Dy was also supported by NSF IIS-0915910 and NIH R01HL089856.

Author Contributions: Jamshid Sourati, Murat Akcakaya, Jennifer Dy, Todd Leen and Deniz Erdogmus conceived the framework and derived the technical equations; Jamshid Sourati and Murat Akcakaya did the analysis of the data and interpreted the results; Jamshid Sourati wrote the manuscript. Jennifer Dy, Todd Leen and Deniz Erdogmus were the senior authors.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix: Update Equation for Multinomial Logistic Regression

In this section, we show that the update equations in Equations (8) and (12), can be efficiently computed for a multinomial logistic regression. More specifically, we see that the gradient $\mathbf{g}_{n+t+1}$ and the inverse Hessian $\mathbf{H}^{-1}_{n+t+1}$ in iteration $t+1$, can be efficiently computed from $\mathbf{g}_{n+t}$ and $\mathbf{H}^{-1}_{n+t}$ in iteration t.

Multinomial Logistic Regression as a Discriminative Classifier

In a classification problem, with $c > 1$ denoting the number of classes, a multinomial logistic regression models the posterior distribution of the class labels given the feature vectors $\mathbf{x}$ and a given $(d+1)(c-1)$-dimensional parameter vector $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top, ..., \boldsymbol{\theta}_{c-1}^\top]^\top$, where $\boldsymbol{\theta}_i = [\alpha_i, \boldsymbol{\beta}_i^\top]^\top$ (for $1 \leq i \leq c-1$), as the following [29]:

$$\begin{aligned} \mathbb{P}(y = j|\mathbf{x}, \boldsymbol{\theta}) &= \frac{e^{\alpha_j + \boldsymbol{\beta}_j^\top \mathbf{x}}}{1 + \sum_{t=1}^{c-1} e^{\alpha_t + \boldsymbol{\beta}_t^\top \mathbf{x}}} \quad , j = 1, ..., c-1 \\ \mathbb{P}(y = c|\mathbf{x}, \boldsymbol{\theta}) &= \frac{1}{1 + \sum_{t=1}^{c-1} e^{\alpha_t + \boldsymbol{\beta}_t^\top \mathbf{x}}} \end{aligned} \tag{24}$$

Now for an indexed feature-label pair $(\mathbf{x}_i, y_i = j)$ define

$$\pi_{ij} := \mathbb{P}(y_i = j|\mathbf{x}_i, \boldsymbol{\theta}) \quad , j = 1, ..., c. \tag{25}$$

Note that π_{ij}, and equivalently the distributions shown in Equation (24), are the likelihood functions when viewed from the point of view of parameter vector $\boldsymbol{\theta}$. The objective to optimize in order to find the maximum likelihood estimation (MLE), denoted by $\hat{\boldsymbol{\theta}}_n$, given an i.i.d set of training samples $\mathcal{L} = \{(\mathbf{x}_1, y_1^*), ..., (\mathbf{x}_n, y_n^*)\}$, is

$$\ell(\boldsymbol{\theta}; \mathcal{L}) \;:=\; \sum_{i=1}^{n} \mathbb{P}(y_i^* | \mathbf{x}_i, \boldsymbol{\theta}) \;=\; \sum_{i=1}^{n} \sum_{j=1}^{c} \mathbb{1}(y_i^* = j) \log \pi_{ij}, \tag{26}$$

that is

$$\hat{\boldsymbol{\theta}}_n \;=\; \arg\max_{\boldsymbol{\theta}} \; \ell(\boldsymbol{\theta}; \mathcal{L}).$$

The subscript n in $\hat{\boldsymbol{\theta}}_n$ is to emphasize the sample size of the training data using which the MLE is obtained.

Updating the Gradient Vector

Here, we formulate the gradient vector in terms of the individual log-likelihood functions $\pi_i j$ and the feature vectors $\mathbf{x}_i$, which readily enables us to derive an update equation for the gradient. The k-th partial gradient (for $1 \leq k \leq c-1$) of the log-likelihood evaluated at $(\mathbf{x}_i, y_i^* = j)$ is:

$$\nabla_k \log \pi_{ij} \;:=\; \frac{\partial \log \pi_{ij}}{\partial \boldsymbol{\theta}_k} \;=\; \Big[\mathbb{1}(j = k) - \pi_{ik}\Big] \cdot \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}, \tag{27}$$

where $\mathbb{1}(\cdot)$ is the indicator function. Since $1 \leq k \leq c-1$, we always get $\mathbb{1}(j = k) = 0$ when $j = c$.

From Eqution (27), we can write the k-th partial gradient of the total log-likelihood function $\ell(\boldsymbol{\theta}; \mathcal{L})$:

$$\nabla_k \ell(\boldsymbol{\theta}; \mathcal{L}) \;=\; \sum_{i=1}^{n} \sum_{j=1}^{c} \mathbb{1}(y_i^* = j) \nabla_k \log \pi_{ij} \;=\; \sum_{i=1}^{n} \Big[\mathbb{1}(y_i^* = k) - \pi_{ik}\Big] \cdot \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}. \tag{28}$$

The complete gradient vector of the log-likelihood function $\ell(\boldsymbol{\theta}; \mathcal{L})$, denoted by $\mathbf{g}_n$, is obtained by concatenating the partial gradient vectors. It can be written compactly as below:

$$\mathbf{g}_n \;=\; \begin{bmatrix} \nabla_1 \ell(\boldsymbol{\theta}; \mathcal{L}) \\ \vdots \\ \nabla_{c-1} \ell(\boldsymbol{\theta}; \mathcal{L}) \end{bmatrix} \;=\; \sum_{i=1}^{n} \begin{bmatrix} \mathbb{1}(y_i^* = 1) - \pi_{i1} \\ \vdots \\ \mathbb{1}(y_i^* = c-1) - \pi_{i1,c-1} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}, \tag{29}$$

where $\otimes$ denotes the Kronecker product. Equation (29) implies that the gradient is additive and the update equation, after adding a pair $(\mathbf{x}_{u_1}, y_{u_1})$ to the training set $\mathcal{L}$ is simply equal to

$$\mathbf{g}_{n+1} \;=\; \mathbf{g}_n \;+\; \begin{bmatrix} \mathbb{1}(y_{u_1} = 1) - \pi_{u_1,1} \\ \vdots \\ \mathbb{1}(y_{u_1} = c-1) - \pi_{u_1,c-1} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \mathbf{x}_{u_1} \end{bmatrix}. \tag{30}$$

Similarly, $\mathbf{g}_{n+t}$ in Equation (12) can be obtained by adding a single product to the gradient vector calculated in the previous iteration:

$$\mathbf{g}_{n+t} \;=\; \mathbf{g}_{n+t-1} \;+\; \begin{bmatrix} \mathbb{1}(y_{u_t} = 1) - \pi_{u_t,1} \\ \vdots \\ \mathbb{1}(y_{u_t} = c-1) - \pi_{u_t,c-1} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \mathbf{x}_{u_t} \end{bmatrix}. \tag{31}$$

Updating the Inverse Hessian Matrix

Let us first focus on calculating the Hessian matrix using the training set $\mathcal{L}$. From Equation (28) and after doing some algebra, the partial second derivative of the log-likelihood function with respect to $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_k$ (for $1 \leq k, s \leq c-1$) is

$$\nabla^2_{sk}\ell(\boldsymbol{\theta};\mathcal{L}) \;:=\; \frac{\partial^2\ell(\boldsymbol{\theta};\mathcal{L})}{\partial\boldsymbol{\theta}_k\partial\boldsymbol{\theta}_s} \;=\; \sum_{i=1}^{n}\pi_{ik}(\pi_{is}-1)\cdot\begin{bmatrix}1\\ \mathbf{x}_i\end{bmatrix}\begin{bmatrix}1 & \mathbf{x}_i^\top\end{bmatrix}. \tag{32}$$

The partial second derivative in Equation (32) form the (s,k)-th block of the total Hessian matrix $\mathbf{H}_n$, which can be written in a compact manner as below:

$$\mathbf{H}_n \;=\; \sum_{i=1}^{n}\boldsymbol{\pi}_i(\boldsymbol{\pi}_i-\mathbf{1})^\top\otimes\begin{bmatrix}1\\ \mathbf{x}_i\end{bmatrix}\begin{bmatrix}1 & \mathbf{x}_i^\top\end{bmatrix}, \tag{33}$$

where $\boldsymbol{\pi}_i = [\pi_{i1},...,\pi_{i,c-1}]^\top$ and $\mathbf{1}$ is a $(c-1)$-dimensional vector with all the elements equal to 1. Therefore, the Hessian matrix also has an additive formulation. The new Hessian matrix $\mathbf{H}_{n+1}$ after adding $(\mathbf{x}_{u_1}, y_{u_1})$ to the training set is equal to

$$\begin{aligned}\mathbf{H}_{n+1} &= \mathbf{H}_n + \boldsymbol{\pi}_{u_1}(\boldsymbol{\pi}_{u_1}-\mathbf{1})^\top\otimes\begin{bmatrix}1\\ \mathbf{x}_{u_1}\end{bmatrix}\begin{bmatrix}1 & \mathbf{x}_{u_1}^\top\end{bmatrix}\\ &= \mathbf{H}_n + \mathbf{R}_1.\end{aligned} \tag{34}$$

Note that the update term $\mathbf{R}_1$ in Equation (34) is the Kronecker product of two rank-one matrices, implying that $\mathbf{H}_{n+1}$ is a rank-one perturbation of $\mathbf{H}_n$. Therefore, we can use Sherman–Morrison inversion formula to write the inverse Hessian update equation:

$$\mathbf{H}_{n+1}^{-1} \;=\; \mathbf{H}_n^{-1} + \frac{\mathbf{H}_n^{-1}\mathbf{R}_1\mathbf{H}_n^{-1}}{1+\operatorname{tr}\left[\mathbf{R}_1\mathbf{H}_n^{-1}\right]}. \tag{35}$$

Hence, we do not need to explicitly perform a matrix inversion in order to update the inverse Hessian matrix. Similarly, for obtaining $\mathbf{H}_{n+t}^{-1}$ from $\mathbf{H}_{n+t-1}^{-1}$ that is needed in Equation (12), we have:

$$\mathbf{H}_{n+t}^{-1} \;=\; \mathbf{H}_{n+t-1}^{-1} + \frac{\mathbf{H}_{n+t-1}^{-1}\mathbf{R}_t\mathbf{H}_{n+t-1}^{-1}}{1+\operatorname{tr}\left[\mathbf{R}_t\mathbf{H}_{n+t-1}^{-1}\right]}, \tag{36}$$

where

$$\mathbf{R}_t \;=\; \boldsymbol{\pi}_{u_t}(\boldsymbol{\pi}_{u_t}-\mathbf{1})^\top\otimes\begin{bmatrix}1\\ \mathbf{x}_{u_t}\end{bmatrix}\begin{bmatrix}1 & \mathbf{x}_{u_t}^\top\end{bmatrix}.$$

References

1. Lewis, D.D.; Gale, W.A. A Sequential Algorithm for Training Text Classifiers. In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, 3–6 July 1994.
2. Freund, Y.; Seung, H.S.; Shamir, E.; Tishby, N. Selective sampling using the query by committee algorithm. *Mach. Learn.* **1997**, *28*, 133–168.
3. Cohn, D.A.; Ghahramani, Z.; Jordan, M.I. Active learning with statistical models. **1996**, arXiv:cs/9603104.
4. Campbell, C.; Cristianini, N.; Smola, A.J. Query Learning with Large Margin Classifiers. In Proceedings of the 17th International Conference on Machine Learning, Stanford, CA, USA, 29 June–2 July 2000.

5. Roy, N.; McCallum, A. Toward Optimal Active Learning through Monte Carlo Estimation of Error Reduction. In Proceedings of the 18th International Conference on Machine Learning, Williamstown, MA, USA, 28 June–1 July 2001.
6. Settles, B.; Craven, M.; Ray, S. Multiple-instance active learning. *Adv. Neural Inf. Process. Syst.* **2008**, *20*, 1289–1296.
7. Brinker, K. Incorporating Diversity in Active Learning with Support Vector Machines. In Proceedings of the 20th International Conference on Machine Learning, Washington, DC, USA, 21–24 August 2003.
8. Holub, A.; Perona, P.; Burl, M.C. Entropy-based active learning for object recognition. In Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Anchorage, AK, USA, 23–28 June 2008.
9. Chen, Y.; Krause, A. Near-optimal batch mode active learning and adaptive submodular optimization. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
10. Guo, Y.; Schuurmans, D. Discriminative batch mode active learning. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.6929 (accessed on 2 February 2016).
11. Azimi, J.; Fern, A.; Zhang-Fern, X.; Borradaile, G.; Heeringa, B. Batch Active Learning via Coordinated Matching. In Proceedings of the 29th International Conference on Machine Learning, Edinburgh, UK, 27 June–3 July 2012.
12. Hoi, S.C.H.; Jin, R.; Zhu, J.; Lyu, M.R. Batch Mode Active Learning and Its Application to Medical Image Classification. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 2006.
13. Guo, Y. Active instance sampling via matrix partition. Available online: http://papers.nips.cc/paper/3919-active-instance-sampling-via-matrix-partition (accessed on 2 February 2016).
14. Li, X.; Guo, Y. Adaptive Active Learning for Image Classification. In Proceedings of 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013.
15. Guo, Y.; Greiner, R. Optimistic Active Learning Using Mutual Information. In Proceedings of 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007.
16. Krause, A.; Singh, A.; Guestrin, C. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.* **2008**, *9*, 235–284.
17. Dilkina, B.; Damoulas, T.; Gomes, C.; Fink, D. AL2: Learning for Active Learning. In Proceedings of Machine Learning for Sustainability Workshop at the 25th Conference of Neural Information Processing Systems, Sirra Nevada, Spain, 16–17 December 2011.
18. Wei, K.; Iyer, R.; Bilmes, J. Submodularity in data subset selection and active learning. In Proceedings of the 32nd International Conference on Machine Learning, Lille, Fran, 6–11 July 2015.
19. Bach, F. Learning with submodular functions: A convex optimization perspective. **2013**, arXiv:1111.6453v2.
20. Nemhauser, G.L.; Wolsey, L.A.; Fisher, M.L. An analysis of approximations for maximizing submodular set functions—I. *Math. Program.* **1978**, *14*, 265–294.
21. Krause, A.; Golovin, D. Submodular Function Maximization. Available online: https://las.inf.ethz.ch/files/krause12survey.pdf (accessed on 2 February 2016).
22. Nemhauser, G.L.; Wolsey, L.A. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.* **1978**, *3*, 177–188.
23. Feige, U.; Mirrokni, V.S.; Vondrak, J. Maximizing non-monotone submodular functions. *SIAM J. Comput.* **2011**, *40*, 1133–1153.
24. Buchbinder, N.; Feldman, M.; Naor, J.; Schwartz, R. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In Proceeding of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, New Brunswick, NJ, USA, 20–23 October 2012; pp. 649–658.
25. Buchbinder, N.; Feldman, M.; Naor, J.; Schwartz, R. Submodular Maximization with Cardinality Constraints. In Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms , Portland, OR, USA, 5–7 January 2014.
26. Lichman, M. UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/ml/datasets.html (accessed on 2 February 2016).
27. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324.

28. Radicioni, D.P.; Esposito, R. BREVE: An HMPerceptron-Based Chord Recognition System. In *Advances in Music Information Retrieval*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 143–164.
29. McCullagh, P.; Nelder, J.A. Generalized linear models. *Ann. Statist.* **1984**, *12*, 1589–1596.