Active Feature Selection for the Mutual Information Criterion

Shachar Schnapp and Sivan Sabato
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel
schnapp@post.bgu.ac.il, sabatos@cs.bgu.ac.il

Abstract

We study active feature selection, a novel feature selection setting in which unlabeled data is available, but the budget for labels is limited, and the examples to label can be actively selected by the algorithm. We focus on feature selection using the classical mutual information criterion, which selects the k features with the largest mutual information with the label. In the active feature selection setting, the goal is to use significantly fewer labels than the data set size and still find k features whose mutual information with the label based on the *entire* data set is large. We explain and experimentally study the choices that we make in the algorithm, and show that they lead to a successful algorithm, compared to other more naive approaches. Our design draws on insights which relate the problem of active feature selection to the study of pure-exploration multi-armed bandits settings. While we focus here on mutual information, our general methodology can be adapted to other feature-quality measures as well. The extended version of this paper, reporting all experiment results, is available at Schnapp and Sabato (2020). The code is available at the following url:
https://github.com/ShacharSchnapp/ActiveFeatureSelection

1 Introduction

Feature selection is the task of selecting a small number of features which are the most predictive with respect to the label. Filter methods for feature selection (see, e.g., Guyon et al. 2008) attempt to select the best features without committing to a specific model or task. We focus here on the classical mutual-information filter method, which selects the features with the largest mutual information with the label.

In the standard feature selection setting, the mutual information of each feature with the label is estimated based on a fully-labeled data set. In this work, we study a novel active setting, in which only the unlabeled data is readily available, and there is a limited budget for labels, which may be obtained upon request. In this setting, the feature selection algorithm can iteratively select an example from the data set and request its label, while taking into account in its decision all the labels obtained so far. The goal is to use significantly fewer labels than the data set size and still find k features whose mutual information with the label based on the *entire* data set is large. This is the analog, for the feature-selection task, of the active learning setting, first proposed in Cohn, Atlas, and Ladner (1994); McCallum and Nigam (1998), which addresses classification tasks.

Feature selection in such an active setting is useful in applications characterized by a preliminary stage of feature selection with expensive labels but readily available features, followed by a deployment stage in which only a few features can be collected for each example. For instance, consider developing a medical protocol which can only use a small number of features, so that patient intake is resource efficient. The goal during protocol design is to select the most informative patient attributes with respect to a target label, such as the existence of a life-threatening condition. In the preliminary study, it is possible to collect a large number of features about a cohort of patients, but identifying the condition of the patient (the label) with a high accuracy may require paying trained specialists. By using an active feature selection algorithm which requires only a small number of labels, the labeling cost at the design stage can be controlled.

As a second example, consider a design/deploy scenario of a computer system which trains classifiers online. The online system handles large volumes, and thus cannot measure all possible features. Therefore, at the design stage, a small number of features must be selected, which will be the only ones collected later by the deployed system. The feature selection process is performed on an offline system, and therefore does not have easy access to labels, but it does allow collecting all the features for the unlabeled feature selection data set, since it handles smaller volumes of data. As a concrete example, consider a web-server which needs to quickly decide where to route client requests, by predicting the resource type that this client session will later use. In the live system, it is easy to collect the true labels downstream and feed them to the training algorithm. However, the server is limited in the number of features it can retrieve for each request, due to bounded communication with the client. On the other hand, in the design stage, there is no limit to communicating with the client, due to the smaller processing volumes. However, simulating the live process in order to identify the label is expensive, since the databases are not local.

Our goal is then to interactively select examples to label from the feature selection data set, and to then use these labeled examples to select k features which have a high mu-

tual information with the label. If the budget for labels is limited, a naive approach is to label a random subset of the unlabeled data set, and to then select the top-ranking features according to this sub-sample. Our main contribution is a practical algorithm that selects the examples to label using a smart interactive procedure. This results in selecting features of a higher quality, that is, features with a larger (true) mutual information with the label. Our design draws on insights relating the problem of active feature selection to the study of pure-exploration of multi-armed bandits (Antos, Grover, and Szepesvári 2008; Carpentier et al. 2011; Chen et al. 2014; Garivier and Kaufmann 2016), which we elaborate on below. While we focus here on mutual information, our general methodology can be adapted to other feature-quality measures, an extension that we leave for future work.

Paper structure. Related work is discussed in Section 2. Section 3 presents the formal setting and notation. In Section 4, we discuss estimating the quality of a single feature is discussed; the solution is an important ingredient in the active feature selection algorithm, which is then presented in Section 5. Experiments are reported in Section 6. We conclude in Section 7. Full experiment results can be found in the full version of the paper (Schnapp and Sabato 2020).

2 Related Work

Feature selection for classification based on a fully labeled sample is a widely studied task. Classical filter methods (see, e.g., Guyon et al. 2008) select features based on a quality measure assigned to each feature. This approach does not take into account possible correlations between features, which is in general a computationally hard problem (Amaldi and Kann 1998). Nonetheless, such filter methods are practical and popular due to their computational tractability. The most popular quality measures estimate the predictive power of the feature with respect to the label, based on the labeled sample. Two popular such measures are mutual information and the Gini index (see, e.g., Hastie, Tibshirani, and Friedman 2001; Guyon et al. 2008; Sabato and Shalev-Shwartz 2008). A different approach is that of the Relief algorithm (Kira and Rendell 1992), which selects features that distinguish similar instances with different labels.

We are not aware of works on feature selection in the active setting studied here. Previous works (Liu et al. 2003; Liu, Motoda, and Yu 2004) propose to use selective sampling to improve the accuracy of the Relief feature selection algorithm within a limited run time. However, they use the labels of the entire data set, and so they do not reduce the labeling cost. Alternatively, one might consider using active learning algorithms which output sparse models to actively select features. The theory of such algorithms has been studied for learning half-spaces under specific distributional assumptions (Zhang 2018). However, we are not aware of practical active learning algorithms for a given sparsity level, which would allow performing joint active learning and feature selection using a limited label budget.

In this work, we show a connection between the active feature selection problem and exploration-only multi-armed bandit settings. A problem of uniformly estimating the means of several random arms with a small number of arm pulls is studied in Antos, Grover, and Szepesvári (2008) and in Carpentier et al. (2011). In Garivier and Kaufmann (2016), a proportion-tracking approach for optimal best-arm identification is proposed. In Chen et al. (2014), the setting of combinatorial pure exploration is studied, in which the goal is to find a subset of arms with a large total quality, using a small number of arm pulls. We discuss these works in more detail in the relevant contexts throughout the paper.

3 Setting and Notations

For an integer i, denote $[i] = \{1, \ldots, i\}$. We first formally define the problem of selecting the top-k features based on mutual information. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the example domain and $\mathcal{Y}$ is the set of labels. Each example $x \in \mathcal{X}$ is a vector of d features $(x(1), \ldots, x(d))$. We assume for simplicity that each feature j accepts one of a finite set of values V_j, and that the label $\mathcal{Y}$ is binary: $\mathcal{Y} = \{0, 1\}$.[1] Denote a random labeled example by $(\mathbf{X}, Y) \sim \mathcal{D}$, where $\mathbf{X} = (X(1), \ldots, X(d))$. The quality of feature j is measured by its mutual information with the label, defined as $I_j := I(X(j); Y)$, equal to

$$\sum_{\substack{y \in \{0,1\} \\ v \in V_j}} \mathbb{P}(X(j)=v, Y=y) \log\left(\frac{\mathbb{P}(X(j)=v, Y=y)}{\mathbb{P}(Y=y)\mathbb{P}(X(j)=v)}\right). \quad (1)$$

Given an integer k, The goal of the feature selection procedure is to select k features with the largest mutual information with the label. In standard (non-active) feature selection, the mutual information of each feature is estimated using a labeled sample of examples. In contrast, in active feature selection, the input is an i.i.d. unlabeled sample $S := (x_1, x_2, ..., x_m)$ drawn according to the marginal distribution of $\mathcal{D}$ over $\mathcal{X}$. At time step t, the algorithm selects an example x_{i_t} from S and requests its label, y_{i_t}, which is drawn according to $\mathcal{D}$ conditioned on $\mathbf{X} = x_{i_t}$. If the label of the same example is requested twice, the same label is provided again. The selection of i_t at time step t may depend on the labels obtained in previous time steps. When the budget of labels is exhausted, the algorithm outputs a set of k features, with the same goal as the standard feature selection algorithm: selecting the features with the largest mutual information with the label.

If the unlabeled sample S is sufficiently large, then the sample distribution of each feature $X(j)$ is sufficiently similar to the true distribution. Moreover, if all the labels for S were known, as in the classical non-active setting, then the plug-in estimator for mutual information, obtained by replacing each probability value in Eq. (1) with its empirical counterpart on the full labeled sample, would be a highly accurate estimator of the true mutual information (see, e.g., Paninski 2003). Therefore, for simplicity of presentation, we assume below that the empirical distribution of each feature on the unlabeled sample is the same as the true distribution, and that the best possible choice of features would be achieved by getting the labels for the entire sample S,

[1] Generalization to multiclass labels is straight-forward; Real-valued features may be quantized into bins.

and then selecting the top-k features according to their mutual information plug-in estimate on the labeled sample. The challenge in active feature selection is to approach this optimal selection as much as possible, while observing considerably fewer labels.

4 Active Estimation for a Single Feature

A core problem for the goal of actively selecting k features based on their quality, is to actively estimate the quality of a single feature. We discuss this problem and its solution, as a stepping stone to the full active feature selection algorithm, which will incorporate this solution. The quality of a single feature j is $I_j = H(Y) - H(Y \mid X(j))$, where H is the entropy function. For $v \in V_j$, denote $p_v := \mathbb{P}[X_j = v]$, $q_v := \mathbb{P}[Y = 1 \mid X(j) = v]$. Since we are interested in estimating I_j only for the purpose of ranking the features, it is sufficient to estimate the conditional entropy:

$$H(Y \mid X(j)) \equiv \sum_{v \in V_j} p_v \cdot H(Y \mid X(j) = v) = \sum_{v \in V_j} p_v \cdot H_b(q_v),$$

where $H_b(q) := q \log(1/q) + (1-q)\log(1/(1-q))$ is the binary entropy. This leads to the single-feature estimation problem, described as follows. We observe m i.i.d. draws $(z_1, \ldots, z_m)$ from the marginal distribution of $\mathcal{D}$ over $X(j)$. At each time step t, the algorithm selects an index i_t and requests the label $y_{i_t} \sim Y|X(j) = z_{i_t}$. The goal is to obtain a good plug-in estimate for $H_j := H(Y|X(j))$ using the available label budget. Since the examples $z_1, \ldots, z_m$ are values from V_j, selecting an index i_t is equivalent to selecting a value in V_j, and we have $y_{i_t} \sim Y|X(j) = v$. As discussed in Section 3, we assume for simplicity of presentation that the data set is sufficiently large, so that p_v for $v \in V_j$ can be obtained by calculating the empirical probabilities of the sample. Thus, the single-feature estimation problem can be reformulated as follows: $\{p_v\}_{v \in V_j}$ are provided as input; at each time step, the algorithm selects a value $v \in V_j$ and obtains one random draw of $Y \mid X(j) = v$. Note that this is possible if the data set is large enough, as we assume. When the label budget is exhausted, H_j is estimated based on the obtained samples. Since $\{p_v\}$ are known, the goal of the active algorithm is to estimate $\{q_v\}$ to a sufficient accuracy. Let $\hat{q}_v$ be the empirical plug-in estimate of q_v, based on the labels obtained for the value v. Then the entropy estimate is $\hat{H}_j := \sum_{v \in V_j} p_v H_b(\hat{q}_v)$. We have thus reduced the single-feature estimation problem to the problem of estimating $\{q_v\}_{v \in V_j}$.

Previous works (Antos, Grover, and Szepesvári 2008; Carpentier et al. 2011) study uniform active estimation of the expectations of several bounded random variables, by iteratively selecting from which random variable to draw a sample. In our problem, each value $v \in V_j$ defines the random variable $Y \mid X(j) = v$, with expected value q_v. In our notation, these works attempt to minimize the ℓ_∞ error measure $\max_{v \in V_j} |q_v - \hat{q}_v|$. Given the true q_v, (Antos, Grover, and Szepesvári 2008) defines the *optimal normalized static allocation*, a vector $\mathbf{w} = (w(v))_{v \in V_j}$ which sums to 1, such that $w(v)$ is the fraction of draws that should be devoted to value $v \in V_j$ to minimize the expected ℓ_∞ error. However, $\mathbf{w}$ depends on the unknown q_v values. The *static-allocation strategy* of Antos, Grover, and Szepesvári (2008) tries to approach the frequencies dictated by $\mathbf{w}$ by calculating an estimate $\hat{\mathbf{w}}$ at each time step, using the plug-in estimate of q_v from the labeled data obtained so far.[2] Let $n^{(t)}(v)$ be the number of labels requested for $v \in V_j$ until time step t. At time step t, the algorithm requests a label for the value v_t which most deviates from the desired fraction of draws as estimated by $\hat{\mathbf{w}}$. Formally, it selects $v_t \in \operatorname{argmax}_{v \in V_j} \hat{w}(v)/n^{(t)}(v)$.

In Carpentier et al. (2011), an improvement to the estimate of $\mathbf{w}$ is proposed, and shown to generate an empirically superior strategy with improved convergence bounds. This strategy replaces each plug-in estimate of q_v in the calculation of $\hat{\mathbf{w}}$ by its *upper confidence bound* (UCB). The UCB of q_v is a value $\hat{q}_v^u$ such that with a high probability over the random samples, $\hat{q}_v^u \geq q_v$. Intuitively, this is more robust, especially when the static allocation strategy which uses the plug-in estimates may stop querying a random variable because of a wrong estimate, and thus never recover.

The approach of Carpentier et al. (2011) can be generalized to minimizing any error measure that depends on q_v and $\hat{q}_v$. We study three approaches for active single-feature estimation of the mutual information, based on three error measures. In Section 6, we report an extensive experimental comparison.

The ℓ_∞ approach. Here, we use the algorithms of Carpentier et al. (2011) for the ℓ_∞ error measure as is, thus attempting to make all estimates accurate. However, this measure ignores the importance of each value, thus many labels could be spent on values v with very small p_v.

The variance approach. Here, we take the weights p_v into account, by replacing the ℓ_∞ error with the variance of the weighted estimate for a budget of B samples:

$$\mathrm{Var}\Big(\sum_v p_v \hat{q}_v\Big) = \sum_v p_v^2 \mathrm{Var}(\hat{q}_v) = \sum_v p_v^2 \frac{q_v(1-q_v)}{B \cdot w(v)}.$$

Minimizing this objective subject to $\sum_v w(v) = 1$ leads to the following static allocation:

$$w_{\mathrm{var}}(v) \propto p_v \sqrt{q_v(1-q_v)}.$$

Here and below, the notation $w(v) \propto \alpha(v)$ for some function α indicates that $w(v) = \alpha(v) / \sum_{u \in V_j} \alpha(u)$. As in the ℓ_∞ approach, we calculate $\hat{w}_{\mathrm{var}}(v)$, the estimate of $w_{\mathrm{var}}(v)$, using UCBs. However, the property of interest here is $q_v(1-q_v)$, and plugging in $\hat{q}_v^u$ instead of q_v does not lead to a UCB on this quantity. Analogously to $\hat{q}_v^u$, let $\hat{q}_v^l$ be a lower bound on the value of q_v, which holds with a high probability. Denote the upper confidence bound of a function f of q_v based on these bounds by [3]

$$\mathrm{UCB}(f, \hat{q}_v^l, \hat{q}_v^u) := \max_{x \in [\hat{q}_v^l, \hat{q}_v^u]} f(x).$$

[2] (Antos, Grover, and Szepesvári 2008) add a small correction term to the plug-in estimate.

[3] If no samples are available for the value v, we set by convention $\hat{q}_v^l = 0$, $\hat{q}_v^u = 1$.

Then, if $q_v \in [\hat{q}_v^l, \hat{q}_v^u]$, we have $f(q_v) \leq \mathrm{UCB}(f, \hat{q}_v^l, \hat{q}_v^u)$. If f is monotonic increasing in $[0, \frac{1}{2}]$ and monotonic decreasing in $[\frac{1}{2}, 1]$, then

$$\mathrm{UCB}(f, \hat{q}_v^l, \hat{q}_v^u) := \begin{cases} f(\frac{1}{2}) & \hat{q}_v^l \leq \frac{1}{2} \leq \hat{q}_v^u, \\ f(\hat{q}_v^l) & \hat{q}_v^l > \frac{1}{2}, \\ f(\hat{q}_v^u) & \text{otherwise } (\hat{q}_v^u < \frac{1}{2}). \end{cases} \quad (2)$$

This holds for $f = f_{\mathrm{var}}$, where $f_{\mathrm{var}}(x) := x(1-x)$. Plugging this into the formula for w_{var} given above, we get $\hat{w}_{\mathrm{var}}(v) \propto p_v \sqrt{\mathrm{UCB}(f_{\mathrm{var}}, \hat{q}_v^l, \hat{q}_v^u)}$. This approach has the advantage of simplicity. However, it is not optimized for our goal, which is to estimate the conditional entropy.

The conditional entropy approach. Here, we set the error to the variance of the conditional entropy estimate: $\mathrm{Var}(\hat{H}_j) \equiv \sum_v p_v^2 \cdot \mathrm{Var}(H_b(\hat{q}_v))$. Since H_b is a smooth function of the random variable $\hat{q}_v$, its variance can be approximated based on its Taylor expansion (Benaroya, Han, and Nagurka 2005). Letting H_b' be the derivative of H_b, and $g(x) := \sqrt{x(1-x)}|\log(\frac{x}{1-x})|$, we have

$$\mathrm{Var}(H_b(\hat{q}_v)) \approx \mathrm{Var}(\hat{q}_v) {H_b'}^2(q_v) = g^2(q_v)/(Bw(v)).$$

Therefore, $\mathrm{Var}(\hat{H}_j) \approx \sum_v p_v^2 g^2(q_v)/(Bw(v))$. Minimizing the RHS subject to $\sum_v w(v) = 1$ gives the static allocation $w_I(v) \propto p_v g(q_v)$. Here too, we estimate $\hat{\mathbf{w}}_I$ using upper confidence bounds. By differentiating g, one can see there is a value $\phi \approx 0.083$ such that g is increasing in $[0, \phi]$ and in $[\frac{1}{2}, 1-\phi]$ and decreasing in $[\phi, \frac{1}{2}]$ and in $[1-\phi, 1]$ (see illustration in Schnapp and Sabato 2020). This implies the following upper confidence bound for g:

$$\mathrm{UCB}(g, \hat{q}_v^l, \hat{q}_v^u) = \begin{cases} g(\phi), \ \phi \in [\hat{q}_v^l, \hat{q}_v^u] \cup [1-\hat{q}_v^u, 1-\hat{q}_v^l] \\ \max\{g(\hat{q}_v^l), g(\hat{q}_v^u)\}, \text{ otherwise.} \end{cases}$$

Replacing $g(q_v)$ in the definition of $w_I(v)$ with its upper confidence bound, we get

$$\hat{w}_I(v) := \frac{\alpha(v)}{\sum_{u \in V_j} \alpha(u)}, \ \alpha(v) := p_v \cdot \mathrm{UCB}(g, \hat{q}_v^l, \hat{q}_v^u). \quad (3)$$

The two new error measures we suggested require calculating $\hat{q}_v^l$ and $\hat{q}_v^u$ for each $v \in V_j$. This can be done using standard closed-form concentration inequalities such as Hoeffding's inequality (Hoeffding 1963) or Bernstein's inequality (Bernstein 1924), as done in Carpentier et al. (2011). However, these bounds can be very loose for small sample sizes. Since $\hat{q}_v$ is Binomially distributed, the tightest concentration bounds are given by the Clopper-Pearson formulas (Clopper and Pearson 1934), which can be numerically calculated. Overall, we consider the following options for the single-feature estimation problem, with their abbreviations given in parenthesis: The two algorithms of Carpentier et al. (2011) for the ℓ_∞ measure, one based on Hoeffding's inequality (MAX-H) and the other on Bernstein's inequality (MAX-B); Static allocation strategy with $\hat{\mathbf{w}}_{\mathrm{var}}$, with each of the concentration inequalities (VAR-H, VAR-B, VAR-CP); Static allocation strategy with $\hat{\mathbf{w}}_I$, with each of the concentration inequalities (I-H, I-B, I-CP). In addition, we consider a naive baseline which allocates the labels proportionally to p_v (PROP). Our experiments, reported in Section 6, show that I-CP is distinctly the most successful alternative. Indeed, it is the most tailored for the estimation task at hand. In the next section, we describe the full active feature selection algorithm, which incorporate the single-feature allocation strategy as a component.

5 The Active Feature Selection Algorithm

The active feature selection algorithm attempts to select k features with the largest mutual information with the label. Naively, one could estimate H_j separately for each of the features using the best single-feature estimation strategy, as described in Section 4. However, this would cause several issues. First, using each label for estimating only a single feature is extremely wasteful. Indeed, the label budget might even be smaller than the number of features. On the other hand, selecting the example to label based on several features may induce a sampling bias. In addition, not all features are as likely to be a part of the output top-k list. This should be taken into account when selecting examples. Our approach mitigates these issues, and indeed performs well in practice, as demonstrated in Section 6.

The general framework is as follows. We define a score $\mathrm{score}(x, F)$ for each example $x \in S$ and feature subset $F \subseteq [d]$. The score estimates the utility of labeling the example x in improving the entropy estimates of the features in F. In each iteration t, we calculate a feature subset $F^{(t)} \subseteq [d]$ based on the information gathered so far, and select an example x, out of those not labeled yet, which maximizes $\mathrm{score}(x, F^{(t)})$. We first discuss the calculation of the subset $F^{(t)}$; thereafter, we present our proposed score. To set $F^{(t)}$, we take inspiration from the problem of Combinatorial Pure Exploration (CPE) of multi-armed bandits Chen et al. (2014). In this problem, the goal is to select a set of reward distributions (arms) that maximizes the total expected reward, by interactively selecting which distribution to sample at each step. A special case of this problem is the top-k arm-selection problem Kalyanakrishnan and Stone (2010), where the selected set of arms must be of size k. The CPE problem does not directly map to the active feature-selection problem, since in CPE, each step provides information about a single arm, while in active feature-selection, each label provides information about all the features. Nonetheless, a shared challenge is to decide which arms/features to consider in each sampling round; That is, how to set $F^{(t)}$.

Our algorithm sets $F^{(t)}$ by adapting the approach of Chen et al. (2014) to active feature selection, as follows: Given an estimator $\hat{H}_j$ for H_j, and high-probability lower and upper bounds $\hat{H}_j^l$ and $\hat{H}_j^u$ for H_j, define two sets of k features. The first set, $F_k^{(t)}$, holds the top-k features according to the estimates $\{\hat{H}_j\}$ (that is, the k features with the smallest $\hat{H}_j$). The second set, $\tilde{F}_k^{(t)}$, holds an alternative choice of top-k features: those with the smallest estimates, when the estimates for $j \in F_k^{(t)}$ are set to $\hat{H}_j^l$, and the estimates for $j \notin F_k^{(t)}$ are set to $\hat{H}_j^u$. $F^{(t)}$ is set to the symmetric difference of $F_k^{(t)}$ and $\tilde{F}_k^{(t)}$. Similarly to Chen et al. (2014), im-

Algorithm 1 AFS: Active Feature Selection for the Mutual Information Criterion

Input: Unlabeled sample $S = (x_1, \ldots, x_m)$, number of features to select k, confidence $\delta \in (0,1)$, label budget B, number of iterations for testing change Λ.

For all $j \in [d], v \in V_j$, $p_v(j) \leftarrow \mathbb{P}_{X\sim S}[X(j) = v]$.
$S_1 \leftarrow ()$; $\forall j \in [d], v \in V_j, n^{(1)}(j,v) \leftarrow 0, \hat{q}_v(j) \leftarrow 0$.
for $t \in [B]$ **do**
  For all $j \in [d]$, $\hat{H}_j \leftarrow \sum_{v\in V_j} p_v(j) H_b(\hat{q}_v(j))$,
  $\hat{H}^l_j \leftarrow \sum_v p_v(j) \cdot \mathrm{LCB}(H_b, \hat{q}^l_v(j), \hat{q}^u_v(j))$,
  $\hat{H}^u_j \leftarrow \sum_v p_v(j) \cdot \mathrm{UCB}(H_b, \hat{q}^l_v(j), \hat{q}^u_v(j))$.
  $F^{(t)}_k \leftarrow$ Top-k features according to $\hat{H}_1, \ldots, \hat{H}_d$
  $\forall j \in F^{(t)}_k, \tilde{H}_j \leftarrow \hat{H}^l_j$. $\forall j \notin F^{(t)}_k, \tilde{H}_j \leftarrow \hat{H}^u_j$.
  $\tilde{F}^{(t)}_k \leftarrow$ Top-k features according to $\tilde{H}_1, \ldots, \tilde{H}_d$.
  $F^{(t)} \leftarrow F^{(t)}_k \triangle \tilde{F}^{(t)}_k$ # The symmetric difference
  if F is empty, **then** break from the loop.
  Calculate $\hat{\mathbf{w}}^{(t)}_I$ according to Eq. (3).
  $i_t \leftarrow \mathrm{argmax}_{x_i \in S\setminus S_t} \mathrm{score}^{(t)}(x_i, F^{(t)})$. # Eq. (4)
  Request the label y_{i_t}.
  $S_{t+1} \leftarrow S_t \circ (x_{i_t}, y_{i_t})$
  for $j \in [d]$ set $v := x_{i_t}(j)$ and **do**
    $n^{(t+1)}(j,v) \leftarrow n^{(t)}(j,v) + 1$
    $\hat{q}_v(j) \leftarrow |\{i \,|\, x_i \in S_t, x_i(j) = v, y_i = 1\}| / n^{(t)}(j,v)$.
    Update $\hat{q}^l_v(j), \hat{q}^u_v(j)$ according to Clopper and Pearson (1934) with confidence parameter δ.
  end for
  if $\sum_{i\in F^{(t)}_k} \hat{H}_i$ has not changed in the last Λ iterations, break from the loop and select the remaining examples to label at random.
end for
Calculate $\hat{H}_j$ for all j based on the labeled examples
Return the top-k features according to $\hat{H}_1, \ldots, \hat{H}_d$.

proving the estimates of these features is expected to have the most impact on the selected top-k set. To calculate the required $\hat{H}_j, \hat{H}^l_j, \hat{H}^u_j$, denote $p_v(j) := \mathbb{P}[X(j) = v]$. Since $H_j = \sum_v p_v(j) H_b(q_v(j))$, we calculate $\hat{H}_j$ by replacing $H_b(q_v(j))$ with $H_b(\hat{q}_v(j))$. For $\hat{H}^u_j$, replace $H_b(q_v(j))$ with $\mathrm{UCB}(H_b, \hat{q}^l_v(j), \hat{q}^u_v(j))$, calculated via Eq. (2). For $\hat{H}^l_j$, replace $H_b(q_v(j))$ with: $\mathrm{LCB}(H_b, \hat{q}^l_v(j), \hat{q}^u_v(j)) := \min_{x\in[\hat{q}^l_v, \hat{q}^u_v]} H_b(x) = \min\{H_b(\hat{q}^l_v(j)), H_b(\hat{q}^u_v(j))\}$.

We now discuss the scoring function $\mathrm{score}(x, F)$. First, we define a score for an example x and a single feature j. $\mathrm{score}(x, F)$ will aggregate these scores over $j \in F$. Denote the set of examples labeled before time step t by S_t, and let $n^{(t)}(j,v) := |\{x \in S_t \mid x(j) = v\}|$ be the number of examples requested so far with value v in feature j. Based on the single-feature static allocation strategy, a naive approach would be to set the score of x for feature j to $\hat{w}^{(t)}_I(j)/n^{(t)}(j, x(j))$. However, this may cause significant sampling bias which could skew the entropy estimates, since the aggregate score may encourage labeling specific combinations of feature values, thereby biasing the estimate of $\{q_v\}$ for some features. While complete bias avoidance cannot be guaranteed in the general case, we propose a computationally feasible heuristic for reducing this bias. We add a correction term to the single-feature score, which rewards feature pairs that have been observed considerably less than their proportion in the data set. We do not consider larger subsets, to keep the heuristic tractable. Denote the sample proportion of examples with values v_1, v_2 for features j_1, j_2 by $\hat{p}(j_1, j_2, v_1, v_2) := \mathbb{P}_{X\sim S}[X(j_1) = v_1, X(j_2) = v_2]$, and the proportion of these pairs in S_t by $\hat{p}^{(t)}(j_1, j_2, v_1, v_2) := \mathbb{P}_{X\sim S_t}[X(j_1) = v_1, X(j_2) = v_2]$. If the ratio between these proportions is large, this indicates a strong sampling bias for this feature pair. Denote $\rho^{(t)} := \hat{p}/\max(\hat{p}^{(t)}, 1/|S|)$.[4] We aggregate the ratios $\rho^{(t)}$ for the relevant pairs of features using an aggregation function denoted ψ, to create a correction term which multiplies the naive single-feature score, thus encouraging labeling examples with under-represented pairs. ψ maps a real-valued vector to a single value that aggregates its coordinate values. A reasonable function should be symmetric and monotonic; Thus, a natural choice is a vector norm, e.g., ℓ_1. Our experiments in Section 6 show that other reasonable choices work similarly well. We define the score for a given function ψ as follows: At time t, the single-feature score of an example x for feature j is:[5]

$$\mathrm{score}^{(t)}(x, j, F^{(t)}) := \frac{\hat{w}^{(t)}(j)}{n^{(t)}(j, x(j)) + 1} \cdot \psi((\rho^{(t)}(j, r, x(j), x(r))_{r\in F^{(t)}\setminus\{j\}}).$$

At time step t, the example with the largest overall score $\mathrm{score}^{(t)}$ is selected, where the score is defined as:

$$\mathrm{score}^{(t)}(x, F^{(t)}) = \psi((\mathrm{score}^{(t)}(x, j, F^{(t)}))_{j\in F^{(t)}}). \quad (4)$$

The full active feature selection algorithm, AFS, is given in Alg. 1. AFS gets as input an unlabeled sample of size m, the number of features to select k, a label budget B, and a confidence parameter δ, which is used to set the lower and upper bounds $\hat{q}^l_v, \hat{q}^u_v$. An additional parameter Λ is used for a safeguard described below. AFS outputs k features which are estimated to have the largest mutual information with the label. The computational complexity of the algorithm is $O(B(d + mk + k\log(d)))$.

AFS includes an additional safeguard, aimed at preventing cases in which the selection strategy of AFS leads to a strongly biased estimate of the mutual information, and the selection strategy itself is too biased to allow collecting labels that correct this estimate. AFS checks if $\sum_{i\in F^{(t)}_k} \hat{H}_i$ for the current top-k features has changed in the last Λ iterations. If it has not changed, AFS selects the remaining examples at random. This guarantees that a wrong estimate will eventually be corrected, while a correct estimate will not be harmed. This safeguard only takes effect in rare cases, but it is important for preventing catastrophic failures.

[4] The maximization circumvents infinity if $\hat{\rho}^{(t)} = 0$.

[5] We add 1 to the denominator to avoid an infinite score; Note that it is impractical in this setting to guarantee one sample of each feature-value pair, since this could exhaust the labeling budget.

6 Experiments

We first report experiments for the single-feature estimation problem, comparing the approaches suggested in Section 4. These clearly show that the I-CP approach is preferable. Then, we report experiments on several benchmark data sets for the AFS algorithm, comparing it to three natural baselines. We further report ablation studies, demonstrating the necessity of each of the mechanisms of the algorithm. Lastly, we compare different choices of ψ. Python 3.6 code for all experiments is available at: https://github.com/ShacharSchnapp/ActiveFeatureSelection. The full experiment results are reported in Schnapp and Sabato (2020). All experiments can be run on a standard personal computer.

Single-feature estimation. We tested the algorithms listed in Section 4 in synthetic and real-data experiments. For the synthetic experiments, we created features with the same p_v for all feature values. This is a favorable setting for MAX-H and MAX-B, which do not take p_v into account. We generated two sets of synthetic experiments, with features of cardinality in $\{2, 4, 6, 8, 10\}$. In the first set, q_v was drawn uniformly at random from $[0, \frac{1}{2}]$; This was repeated 5 times for each set size, resulting in 25 experiments. In the second set, we tested cases with extreme q_v values. For each set size, we set n values to have $q_v = 1/2$ and the rest to $q_v = \alpha$, for all combinations of $\alpha \in \{0.1, 0.01, 0.001\}$ and $n \in \{0, 1, \ldots, |V_j|\}$. This resulted in 95 synthetic experiments. For the real-data experiments, we tested the 14 features of the `Adult` data set (Dua and Graff 2019) with their true p_v values. We ran each test scenario 10,000 times and calculated the average estimation error, defined as $\text{err} := |\hat{H}_j - H_j|$. We further calculated 95% student-t confidence intervals. For algorithm i, denote its confidence interval $[\text{err}_i^l, \text{err}_i^u]$. We say that i has a "clear win" if $\text{err}_i^u \leq \min_{j \neq i} \text{err}_j^l$, and a "win" if $\text{err}_i^l \leq \min_{j \neq i} \text{err}_j^u$. I-CP was the most successful of all algorithms. Table 2 reports numbers of clear wins and wins for the more successful algorithms. The full set of results is available at (Schnapp and Sabato 2020).

Active feature selection. Our active feature selection experiments were performed on all the large data sets with binary labels and discrete features in the feature-selection repository of Li et al. (2018). In addition, we tested on the MUSK data set (Dua and Graff 2019), and on the MNIST data set (LeCun and Cortes 2010) for three pairs of digits. Data set properties are listed in Table 1. We tested feature numbers $k \in \{1, 5, 10, 20\}$.

We compared our algorithm to three natural baselines, which differ in how they select the examples to label. In all baselines, the selected k features were the ones with the largest mutual information estimate, calculated based on the selected labels. The tested baselines were: (1) RANDOM: Select the examples to label at random from the data set; This is the passive baseline, since it is equivalent to running passive feature selection based on the mutual information criterion on a random labeled sample of size B. (2) CORESET: Use a coreset algorithm to select the B most representative examples from the data set. We used the classical Farthest-First Traversal algorithm (see, e.g., Ros and Guillaume 2017, 2020) with the Hamming distance, which is the most appropriate for categorical values. (3) DWUS: Label the examples selected by the active learning algorithm Density Weighted Uncertanty Sampling (Nguyen and Smeulders 2004), implemented in the python package `libact` (Yang et al. 2017).

Data set	Instances	Features	Classes
BASEHOCK	1993	4862	2
PCMAC	1943	3289	2
RELATHE	1427	4322	2
MUSK	6598	167	2
MNIST: 0 vs 1	14,780	784	2
MNIST: 3 vs 5	13,454	784	2
MNIST: 4 vs 6	13,700	784	2

Table 1: Data set properties

Budget	PROP	MAX-B	VAR-H	I-CP
50	(0, 12)	(0, 3)	(0, 12)	$(\mathbf{10}, \mathbf{23})$
100	(0, 4)	(0, 4)	(0, 4)	$(\mathbf{21}, \mathbf{25})$
300	(0, 4)	(0, 4)	(0, 4)	$(\mathbf{15}, \mathbf{25})$
500	(0, 3)	(0, 3)	(0, 3)	$(\mathbf{9}, \mathbf{25})$
50	(0, 21)	(0, 6)	(0, 21)	$(\mathbf{40}, \mathbf{68})$
100	(0, 13)	(0, 8)	(0, 13)	$(\mathbf{48}, \mathbf{69})$
300	(0, 17)	(0, 5)	(0, 17)	$(\mathbf{39}, \mathbf{71})$
500	(0, 15)	(0, 6)	(0, 15)	$(\mathbf{43}, \mathbf{78})$
50	(0, 11)	(0, 1)	(0, 11)	$(\mathbf{2}, \mathbf{13})$
100	(0, 5)	(0, 0)	(0, 5)	$(\mathbf{6}, \mathbf{14})$
300	(0, 3)	(0, 1)	(0, 3)	$(\mathbf{6}, \mathbf{14})$
500	(0, 5)	(0, 1)	(0, 5)	$(\mathbf{5}, \mathbf{13})$

Table 2: Single-feature experiments, reporting numbers of (clear wins, wins) for each algorithm. Top to bottom: synthetic, uniform q_v; synthetic, fixed q_v; The `Adult` data set.

To compare the algorithms, we calculated for each algorithm the mutual information gap between the true top-k features and the selected features, calculated via $\sum_{j \in F^*} H_j - \sum_{j \in F} H_j$, where F are the top-k features based on the collected labels, and F^* are the top-k features according to the true H_j, which was calculated from the full labeled sample.

We ran each experiment 30 times. All graphs plot the average score with 95% confidence intervals. We provide in Figure 1 the graphs for three of the experiments, demonstrating the advantage of AFS over the baselines. In all experiments, AFS performs better or comparably to the best baseline; its improvement is larger for larger values of k.

Ablation tests. We studied the necessity of the mechanisms used by AFS, by testing the following variations:

1. Select the example to label by the score of a single randomly-selected feature (SINGLE);
2. Select the example with the largest average score over all features, without a bias correction term (AVG-ALL);

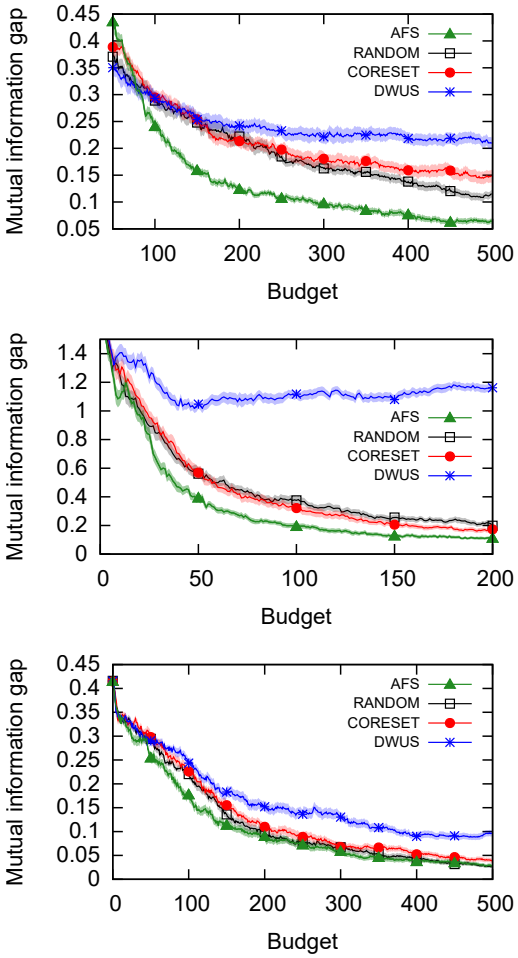

Figure 1: Some of the experiments comparing to baselines. Top: MUSK, $k = 20$. Middle: MNIST, 3 vs 5, $k = 20$. Bottom: RELATHE, $k = 10$. See full experiment results in (Schnapp and Sabato 2020).

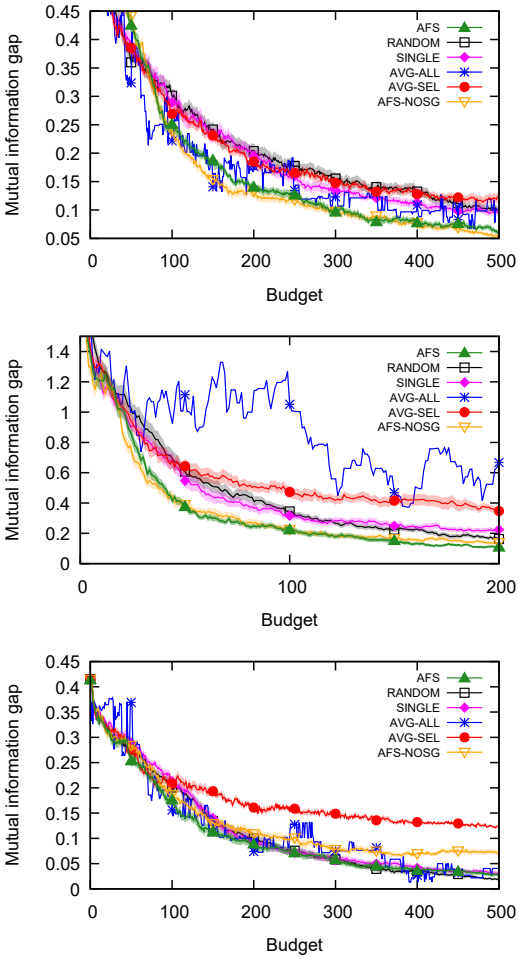

Figure 2: Some of the ablation tests. Top: MUSK, $k = 20$. Middle: MNIST, 3 vs 5. Bottom: RELATHE, $k = 10$. See full experiment results in (Schnapp and Sabato 2020).

3. Select the example with the largest average score over $F^{(t)}$, without a bias correction term (AVG-SEL);

4. The full AFS, with $\psi = \ell_1$, $\delta = 0.05$, but without the safeguard (that is, $\Lambda = \infty$) (AFS-NOSG);

5. The full AFS, with $\psi = \ell_1, \delta = 0.05, \Lambda = 30$ (AFS).

The value of Λ for AFS was selected after testing the values $10, 20, 30, 40$. The five variations above were tested on all the data sets in Table 1 for $k \in \{5, 10, 20\}$. We provide in Figure 2 the graphs for three of the ablation tests, which show that AFS is the only algorithm that performs consistently well. We further compared our choice of $\psi := \ell_1$ to other natural options: $\psi := \ell_\infty$ and $\psi := \ell_2$. We observed (see Schnapp and Sabato 2020) that in most cases, the results are similar for ℓ_1 and ℓ_2, while ℓ_∞ is sometimes slightly worse. Thus, while we chose ℓ_1, setting $\psi := \ell_2$ is also a valid choice.

7 Conclusion

We showed that actively selecting examples to label can improve the performance of feature selection for the mutual information criterion under a limited label budget. We studied the challenges involved in designing such an algorithm, and provided AFS, which improves the quality of the selected features over baselines. In future work, we will study adaptations of the suggested approach for other quality measures, such as the Gini index, and variants of the mutual information criterion that take feature correlations into account.

References

Amaldi, E.; and Kann, V. 1998. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science* 209(1-2): 237–260.

Antos, A.; Grover, V.; and Szepesvári, C. 2008. Active learning in multi-armed bandits. In *International Conference on Algorithmic Learning Theory*, 287–302. Springer.

Benaroya, H.; Han, S. M.; and Nagurka, M. 2005. *Probability models in engineering and science*. CRC press.

Bernstein, S. N. 1924. On a modification of Chebyshev's inequality and of the error formula of Laplace. *Annals Science Institute Sav. Ukraine* Sect. Math. 1.

Carpentier, A.; Lazaric, A.; Ghavamzadeh, M.; Munos, R.; and Auer, P. 2011. Upper-confidence-bound algorithms for active learning in multi-armed bandits. In *International Conference on Algorithmic Learning Theory*, 189–203. Springer.

Chen, S.; Lin, T.; King, I.; Lyu, M. R.; and Chen, W. 2014. Combinatorial pure exploration of multi-armed bandits. In *Advances in Neural Information Processing Systems*, 379–387.

Clopper, C. J.; and Pearson, E. S. 1934. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika* 26(4): 404–413.

Cohn, D.; Atlas, L.; and Ladner, R. 1994. Improving generalization with active learning. *Machine Learning* 15: 201–221.

Dua, D.; and Graff, C. 2019. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml. Date accessed: 2020-09-20.

Garivier, A.; and Kaufmann, E. 2016. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, 998–1027.

Guyon, I.; Gunn, S.; Nikravesh, M.; and Zadeh, L. A. 2008. *Feature extraction: foundations and applications*, volume 207. Springer.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning*. Springer.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* 58(301): 13–30.

Kalyanakrishnan, S.; and Stone, P. 2010. Efficient Selection of Multiple Bandit Arms: Theory and Practice. In *ICML*, volume 10, 511–518.

Kira, K.; and Rendell, L. A. 1992. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, 129–134.

LeCun, Y.; and Cortes, C. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. URL http://yann.lecun.com/exdb/mnist/. Date accessed: 2020-09-20.

Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R. P.; Tang, J.; and Liu, H. 2018. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)* 50(6): 94. URL {http://featureselection.asu.edu}.

Liu, H.; Motoda, H.; and Yu, L. 2004. A selective sampling approach to active feature selection. *Artificial Intelligence* 159(1-2): 49–74.

Liu, H.; Yu, L.; Dash, M.; and Motoda, H. 2003. Active feature selection using classes. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 474–485. Springer.

McCallum, A. K.; and Nigam, K. 1998. Employing EM and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, 359–367. Citeseer.

Nguyen, H. T.; and Smeulders, A. 2004. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, 79.

Paninski, L. 2003. Estimation of entropy and mutual information. *Neural computation* 15: 1191–1253.

Ros, F.; and Guillaume, S. 2017. DIDES: a fast and effective sampling for clustering algorithm. *Knowledge and information systems* 50(2): 543–568.

Ros, F.; and Guillaume, S. 2020. *Sampling Techniques for Supervised Or Unsupervised Tasks*. Springer.

Sabato, S.; and Shalev-Shwartz, S. 2008. Ranking Categorical Features Using Generalization Properties. *Journal of Machine Learning Research* 9: 1083–1114.

Schnapp, S.; and Sabato, S. 2020. Active Feature Selection for the Mutual Information Criterion. Available as arXiv preprint https://arxiv.org/abs/2012.06979.

Yang, Y.-Y.; Lee, S.-C.; Chung, Y.-A.; Wu, T.-E.; Chen, S.-A.; and Lin, H.-T. 2017. libact: Pool-based Active Learning in Python. Technical report, National Taiwan University. URL https://github.com/ntucllab/libact. Available as arXiv preprint https://arxiv.org/abs/1710.00379.

Zhang, C. 2018. Efficient active learning of sparse halfspaces. *arXiv preprint arXiv:1805.02350* .