Intelligently Querying Incomplete Instances for Improving Classification Performance

Karthik Sankaranarayanan
kartsank@in.ibm.com
Human Language Tech. Dept.
IBM India Research Lab
Outer Ring Road
Bangalore, India

Amit Dhurandhar
adhuran@us.ibm.com
Mathematical Sciences Dept.
IBM T.J. Watson
1101 Kitchawan Road
Yorktown Heights, USA

ABSTRACT

The problem of intelligently acquiring missing input information given a limited number of queries to enhance classification performance has gained substantial interest in the last decade or so. This is primarily due to the emergence of the targeted advertising industry, which is trying to best match products to its potential consumer base in the absence of complete consumer profile information. In this paper, we propose a novel active feature acquisition technique, to tackle this problem of instance completion prevalent in these domains. We show theoretically that our technique is optimal given the current classifier and derive a probabilistic lower bound on the error reduction achieved with our technique. We also show that a simplification of our technique is equivalent to the expected utility approach, which is one of the most sophisticated solutions for this problem in existing literature. We then demonstrate the efficacy of our approach through experiments on real data. Finally, we show that our technique can be easily extended to the scenario where we have a cost matrix associated with acquiring missing information for each instance or instance-feature combinations.

Categories and Subject Descriptors

H.2.8 [**Knowledge Management**]: Data Mining

General Terms

Classification

Keywords

Classification, Instance completion, Missing values

1. INTRODUCTION

In this day and age of customization and personalization of products, targeted advertising has gained prominence with the emergence of many start-ups focused solely on selecting the right consumer base that will buy certain products. One of the main challenges faced here is having to deal with incomplete customer information. Hence, while you might be able to build classification models by associating available demographic features to consumer buying decisions, such classification models are usually far from perfect. For example, we may not know the age and/or gender and/or address of some customers, which might be critical in determining their preference for a certain product. The goal then is to improve our understanding of their behaviors by obtaining this missing information. One way in which this is tackled is by identifying a subset of individuals to target so as to maximize the improvement in classification keeping in mind the constraints associated with obtaining such information (budget costs, number of calls you can make, etc.). This problem is not just limited to the targeted advertising industry, but is also relevant to institutions such as government agencies who conduct surveys to understand social tendencies or behaviors. In such cases identifying and contacting the right people can lead to greatly improved understanding of relevant social dynamics at a manageable cost.

More formally, the problem addressed in this paper can be stated as follows: *Given that we can query k instances from a dataset of size N having incomplete input information but known outputs, which instances when queried would maximize the classification performance?* In the application domains described before, this would correspond to choosing k people to call that will most likely significantly enhance classification accuracy having obtained this additional information. This problem has been introduced before in [12] and is commonly referred to as instance completion [13, 7, 19].

It is easy to see that this problem setting is quite different from the traditional active learning problem [16, 2], which involves the complementary problem of missing *outputs* with all inputs known with the goal now being to query those outputs that can maximize classification performance. Other related work includes work on budgeted learning [10, 8], where one is trying to find which entry (instance-feature combination) to query given a certain cost constraint. Knowledge gradient methods [4] are a different class of methods which are also trying to find the optimal entry to query that will maximize the gain in information. However, both these classes of methods are suited for a different setting than ours.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505570 ...$15.00.

Budgeted learning and knowledge gradient methods are generally modeled as markov decision processes (MDPs), and as mentioned before, are trying to find which entry to query rather than a set of entries comprising of multiple instances. To determine the reward in querying an entry, an optimization problem has to be solved for each entry using these methods, which can be expensive. Moreover, even if one were to extend these methods in a straightforward manner for instance completion (where we want to choose k instances out of a possible N), there would be $\frac{N!}{k!(N-k)!}$ possible actions[1] for the corresponding MDP to choose from, which would make this framework computationally infeasible.

In our setting, we seek to maximize the classification performance on a dataset during model building (at training time). However, there is work on active feature acquisition with the objective of getting the best possible performance during model application (at testing time) [7, 1, 17]. In [7], the authors try to query instances in the test set so as to minimize classification uncertainty, while in [1, 17] the goal is to query test instances so as to select the most important features. Clearly, both these lines of work address a goal different from ours.

The most relevant work that addresses the same problem as ours is [19, 12, 13]. In the work of [19], a method called Goal Oriented Data Acquisition (GODA) is described, which initially builds a model based only on complete instances. It then rebuilds a model for each incomplete instance that is completed based on imputed values, by adding it to the complete instances and choosing those incomplete instances that give maximum improvement in performance. In [12], a method called Error Sampling (ES) is described for pick the k instances to query which works as follows: If there are m misclassified instances with missing values and if $m > k$, then k of these m instances are randomly picked. Otherwise if $m \leq k$, then after picking all of these m instances, the remaining $k - m$ are randomly chosen from the correctly classified instances based on an uncertainty score [9, 15]. Not only was the accuracy obtained by ES shown to be better than GODA, but GODA was also computationally much more expensive as one has to first impute all missing values and then retrain a classifier as many times as the number of incomplete instances. In [13], the authors provide a method to score each missing *entry* (instance-feature combination) based on the calculation of Expected Utility (EU). It is easy to adapt this scheme to the task of instance completion by simply adding the expected utilities for the missing values corresponding to each instance and obtain a score.

One of the conceptual deficiencies of ES is that it does not rank instances, especially the misclassified ones, based on any criteria that is directly related to the goal of correctly classifying them, and therefore, most likely to improve classification performance. Even the correctly classified instances are picked based on the classifier margin, which is unrelated to the likelihood that the chosen instance would improve classification upon completion. Addressing this deficiency is the primary motivation behind our technique, and we therefore propose an approach to rank the incomplete instances by directly calculating the probability of correctly classifying each instance given the current classifier based on a function of all the corresponding missing entries. We compute the probability by taking into account all of missing features for an instance together, rather than looking at them independently. *An important fact to notice here is that though there are multiple missing features for each instance, since we are computing a function over all of them, we only need to employ a univariate probability estimation technique instead of a multivariate approach, which makes our method efficient.* A further motivation for choosing instances to query with high probability of being correctly classified is that we learn useful actionable information about our data once they are completed. In particular, if we pick instances by our method to query, then after completion they may be correctly or incorrectly classified. If most of them are correctly classified, then we can be reasonably confident that the classifier we have is reliable. If they aren't, then that implies that our current data is not representative of the underlying data generation process and that we either need to query many more instances and/or collect additional discriminative features. On the other hand, if we had picked instances with high misclassification probability to query, which is diametric to our strategy, then in either case we do not learn much in terms of improving performance. This is because, if most of them are incorrectly classified after completion, then that just confirms what we already knew, while if most of them are correctly classified after completion, then that simply tells us that the problem is in all likelihood easier than we thought.

It is also important to note that in the EU method, the expected utility score for an instance is based on marginals as the expected utility is computed independently for each missing entry. Hence, even though the decision based on their method is optimal for an entry given the current classifier, it isn't optimal for the instance. It turns out that computing the expected utility for each entry can be computationally intensive and they therefore suggest randomly choosing a subset, which makes the method sub-optimal even when choosing which entries to query. Moreover, their method is restricted to categorical input features. We demonstrate that our strategy is optimal at the instance level, and applies to continuous as well as categorical features. We also show that a simplification of our method restricted to categorical features and assuming independence is in fact equivalent to the EU method.

The rest of the paper is organized as follows. In Section 2, we describe in detail the proposed method and show how it would apply in the context of some commonly used classification methods. In Section 3, we show that our strategy is optimal given the current classifier. We then derive a probabilistic lower bound on the error reduction for the optimal classifier on the updated dataset obtained after querying. In Section 4, we describe a simplified univariate version of our method and show that it is equivalent to the EU method when restricted to categorical features. In Section 5, we empirically show that our method is robust and outperforms other competing methods under varying levels of label noise. In Section 6, we describe an easy way to extend our method if a cost matrix is available and discuss promising directions for future research.

2. METHODOLOGY

Before describing our method, we introduce some notation. Let X, Y denote the input and output spaces respectively and $D_N = \{(x_1, y_1), ..., (x_N, y_N)\}$ denote a dataset of size N containing missing input values. Let $\zeta : X \rightarrow Y$ be

[1]! denotes factorial.

Algorithm 1 The proposed method (JIS) to choose k instances to query so as to maximize classification performance.

Input: D_N, k, ζ
Output: D_Q {dataset of k instances to query}
Let $Q = \phi$
if $m > k$ {Number of misclassified and incomplete instances is $> k$.} **then**
  for all $x \in D_m$ **do**
    Compute $f(x^{M(x)}) \in S_x$ where $x^{M(x)} \in V_x$ such that, $\forall\ v \in V_x$, $\zeta(x^{T(x) \cup v}) = y_x$ {Finding the criteria (or function) the missing features have to satisfy to correctly classify x into y_x based on ζ.}
    Estimate $score_x = P(f(x^{M(x)}) \in S_x)$
    $Q = Q \cup score_x$
  end for
  Let Q_k contain the highest k scores in Q
  Let $D_Q = \{x \in D_m | score_x \in Q_k\}$
else
  Let $D_Q = D_m$
  if $m < k$ {Number of misclassified and incomplete instances is $< k$.} **then**
    for all $x \in D_r$ **do**
      Compute $f(x^{M(x)}) \in S_x$ where $x^{M(x)} \in V_x$ such that, $\forall\ v \in V_x$, $\zeta(x^{T(x) \cup v}) = y_x$ {Finding the criteria (or function) the missing features have to satisfy to keep x in the correct class y_x based on ζ.}
      Estimate $score_x = P(f(x^{M(x)}) \in S_x)$
      $Q = Q \cup score_x$
    end for
    Let Q_{k-m} contain the highest $k - m$ scores in Q
    Let $D_Q = D_Q \cup \{x \in D_r | score_x \in Q_{k-m}\}$
  end if
end if
Return D_Q

the optimal classifier (the classifier with the highest accuracy) learned over D_N. Let $D_m \in D_N$ be the set of $m \leq N$ instances having missing information misclassified by $\zeta(.)$ while D_r denote the set of r instances with missing information correctly classified by $\zeta(.)$. Let $T(x_i)$ and $M(x_i)$ denote the set of features with known values and missing values for instance x_i respectively. Correspondingly, let $x_j^{T(x_i)}$ and $x_j^{M(x_i)}$ denote the set of entries i.e. features in x_j, whose values are known for x_i and missing for x_i respectively.

2.1 Method Description

Algorithm 1 elucidates our strategy for choosing the k instances to query. Similar to some prior studies, we first focus on the misclassified instances because trying to correct them could have a greater impact on the final accuracy than those that are already correctly classified. However, it is strategy of deciding exactly which k instances to pick that is critical to the success of this task and is therefore the novel contribution of our method.

The simplest scenario is where $m = k$ in which case we simply choose all the instances from D_m to query.

Now if $m > k$, then for each misclassified instance x in D_m we compute a function of the missing features and the corresponding set of values S_x, such that if the function takes values in the set S_x, which implies the missing features would take values in some other set V_x, then x gets classified into the correct class by ζ. Essentially what we are trying to find is based on the current inductive function ζ, what set of values for a function of the missing features would result in ζ correctly classifying the instance (the details of how this function is computed for some popular classification techniques is described in the next section).

The criteria $f(M(x)) \in S_x$ for any x can be viewed as a *sufficient* condition where the total error (number of misclassified instances) reduces at least by 1 for an updated classifier that is trained on D_N but now with completed instance x (i.e. $x = x^{T(x)}$). This is explained in detail in the section 3 where we discuss error reduction.

The next step in our algorithm is to compute the probability of this function having values in the corresponding set. This probability signifies the likelihood of the instance to be classified correctly after querying it given the current classifier, and can be estimated using common parametric (normal density, exponential, etc.) or non-parametric (histograms, parzen windows [6]) density estimation techniques based on the already known values. One could also use probabilistic inequalities such as Markov inequality or Chebyschev's inequality [5] to get fast upper bounds on the desired probability.

An important fact to note here is that *even though we can have multiple missing features, since we are directly computing a function over all of them, we simply need to employ a univariate probability estimation on the values that the function takes.* This important observation makes the procedure tractable, which would not have been the case if we were to find the appropriate ranges for each missing feature separately that would correctly classify the instance, and then employ a joint probability estimation procedure.

The probabilities of these functions are used as scores for these instances and using them we query the top k instances with the highest scores. Since we compute the probability of a function of the missing features considering them together rather than analyzing them independently, we refer to our method as Joint Instance Selection (JIS).

In the case where $m < k$, then we first choose all of D_m instances to query. For choosing the remaining $k-m$ instances from the set D_r (instances that are correctly classified having missing values), the strategy is similar to the previous case. We first compute the function with values for it that will *keep the instance correctly classified.* We then compute the probability for this function to have those values giving us the score for that instance. We now choose the top $k-m$ instances sorted by their scores, which along with D_m result in the k instances we choose to query.

2.2 Example Instantiations

In the previous subsection, we discussed our general strategy to choose the k instances to query. An integral part of this strategy is to compute a function of the missing features satisfying certain conditions that would result in correctly classifying the instance. In this subsection, we show how this function could be computed for some popular classification techniques.

2.2.1 Linear and Non-linear SVM

Support vector machines (SVMs) [18] are one of the most commonly used classification techniques in academia and industry. They are considered to be quite robust and can be

used to perform linear and non-linear classification. Non-linear classification is accomplished by employing the kernel trick. If we denote the kernel function by $\mathcal{K}(.,.)$ and if α_i denotes the dual variable in SVM optimization corresponding to instance x_i obtained by learning over D_N which gives us the classifier ζ, then the correct classification for x_i is encoded in the following well known inequality,

$$y_i \sum_{j=1}^{N} \alpha_j y_j \mathcal{K}(x_j, x_i) \geq 1 \tag{1}$$

where, $Y = \{1, -1\}$. Without loss of generality (w.l.o.g.) assume that $y_i = 1$, then the above equation can be written as,

$$\begin{aligned} &\sum_{j=1}^{N} \alpha_j y_j \mathcal{K}(x_j, x_i) \geq 1 \\ &\sum_{j=1}^{N} \alpha_j y_j \mathcal{K}(x_j^{T(x_i)} \cup x_j^{M(x_i)}, x_i^{T(x_i)} \cup x_i^{M(x_i)}) \geq 1 \\ &f(x_i^{M(x_i)}) \geq 1 \end{aligned} \tag{2}$$

The reason the left hand side of the above equation can be written as simply a function of the missing features in x_i is that, for the other missing values we can substitute the values used while building the classifier ζ (which could be class conditional means, modes, etc.). To demonstrate this, we next look at some commonly used kernel functions and further clarify our approach.

Linear SVM: In case of Linear SVMs the kernel function is given as,

$$\begin{aligned} \mathcal{K}(x_j, x_i) &= x_j \cdot x_i \\ &= x_j^{T(x_i)} \cdot x_i^{T(x_i)} + x_j^{M(x_i)} \cdot x_i^{M(x_i)} \end{aligned} \tag{3}$$

Using the above equation along with Eqn.2 we have,

$$\begin{aligned} &\sum_{j=1}^{N} \alpha_j y_j [x_j^{T(x_i)} \cdot x_i^{T(x_i)} + x_j^{M(x_i)} \cdot x_i^{M(x_i)}] \geq 1 \\ &f(x_i^{M(x_i)}) \geq 1 \end{aligned} \tag{4}$$

Even though this could be further simplified by moving the terms with $T(x_i)$ (known values) to the right hand side leaving only terms with $M(x_i)$ on the left, we avoid doing so in order to keep the approach consistent because such further simplifications may no necessarily be possible for other kernel functions, as we shall see next.

SVM with Polynomial Kernel: In case of Polynomial Kernel SVMs the kernel function is given as,

$$\begin{aligned} \mathcal{K}(x_j, x_i) &= (x_j \cdot x_i)^b \\ &= (x_j^{T(x_i)} \cdot x_i^{T(x_i)} + x_j^{M(x_i)} \cdot x_i^{M(x_i)})^b \end{aligned} \tag{5}$$

Employing this in Eqn.2 we get,

$$\begin{aligned} &\sum_{j=1}^{N} \alpha_j y_j \left(x_j^{T(x_i)} \cdot x_i^{T(x_i)} + x_j^{M(x_i)} \cdot x_i^{M(x_i)} \right)^b \geq 1 \\ &f(x_i^{M(x_i)}) \geq 1 \end{aligned} \tag{6}$$

Notice that a further simplication to obtain an explicit function of $x_i^{M(x_i)}$ independent of the $T(x_i)$ terms on the left hand side isn't possible. However, this is immaterial as this is still an implicit function of $x_i^{M(x_i)}$.

SVM with Radial Basis Kernel: In this case, the kernel function is given as $\mathcal{K}(x_j, x_i) = e^{-\gamma ||x_j - x_i||^2}$, where $\gamma > 0$. Given this and equation 2 we have,

$$\begin{aligned} &\sum_{j=1}^{N} \alpha_j y_j e^{-\gamma ||x_j - x_i||^2} \geq 1 \\ &\sum_{j=1}^{N} \alpha_j y_j e^{-\gamma ||x_j^{T(x_i)} - x_i^{T(x_i)}||^2} e^{-\gamma ||x_j^{M(x_i)} - x_i^{M(x_i)}||^2} \geq 1 \\ &f(x_i^{M(x_i)}) \geq 1 \end{aligned} \tag{7}$$

Similar to the earlier case, this again is an implicit function of $x_i^{M(x_i)}$.

2.2.2 Logistic Regression

Logistic regression [11] uses the logit link function to relate an input x_i to its output $Y = \{0, 1\}$ as follows,

$$ln \left(\frac{P(y_i = 1)}{1 - P(y_i = 1)} \right) = x_i \cdot \beta \tag{8}$$

where β is the learned parameter vector over D_N. x_i is classified into class 1 iff $P(y_i = 1) \geq 0.5$. Given this and w.l.o.g. assuming $y_i = 1$ the criteria to classify x_i into the correct class is,

$$e^{x_i \cdot \beta} \geq 1 \tag{9}$$

Further expressing x_i in terms of $T(x_i)$ and $M(x_i)$ we get,

$$\begin{aligned} e^{x_i^{T(x_i)} \cdot \beta^{T(x_i)} + x_i^{M(x_i)} \cdot \beta^{M(x_i)}} &\geq 1 \\ e^{x_i^{T(x_i)} \cdot \beta^{T(x_i)}} e^{x_i^{M(x_i)} \cdot \beta^{M(x_i)}} &\geq 1 \\ e^{x_i^{M(x_i)} \cdot \beta^{M(x_i)}} &\geq \frac{1}{e^{x_i^{T(x_i)} \cdot \beta^{T(x_i)}}} \\ f(x_i^{M(x_i)}) &\geq \frac{1}{e^{x_i^{T(x_i)} \cdot \beta^{T(x_i)}}} \end{aligned} \tag{10}$$

Here we were able to obtain an exclusive function of $x_i^{M(x_i)}$ on the left hand side.

These instantiations demonstrate how our strategy can be employed for standard classification techniques. We next proceed to examine the performance of the proposed method in terms of its optimality and bounds on error reduction.

3. PERFORMANCE ANALYSIS

In this section we prove two results. First we show that given the current information (i.e. the current classifier ζ and the input-output), the proposed method is the optimal strategy for minimizing the expected error. We then derive lower bounds on probability of total error reduction in the range of $\{1, ..., min(m, k)\}$.

3.1 Optimality

W.l.o.g. assume that the first $m \leq N$ instances denoted by D_m in D_N are misclassified by $\zeta(.)$ and have missing information. If $\lambda : \mathcal{N} \times \mathcal{N} \rightarrow \{0, 1\}$ denotes the classification loss function, then the total error ϵ is given by,

$$\begin{aligned} \epsilon &= \sum_{i=1}^{N} \lambda(\zeta(x_i), y_i) \\ &= \sum_{i=1}^{m} \lambda(\zeta(x_i), y_i) + \sum_{i=m+1}^{N} \lambda(\zeta(x_i), y_i) \end{aligned} \tag{11}$$

The task is to query those k instances which will minimize this error. If $m \geq k$ the best we can hope to reduce the error by is k which would occur if after querying, the k misclassified points get correctly classified. If $m < k$, then the maximum reduction in error we can expect is m, which would require querying all m of these instances.

Formally, if D_s denotes a dataset of size s then,

$$\begin{aligned} &\underset{D_k \in D_N}{\operatorname{argmin}}\ \epsilon \\ &= \underset{D_k \in D_N}{\operatorname{argmin}} \left(\sum_{\substack{(x_i,y_i) \\ \in D_k}} \lambda(\zeta(x_i), y_i) + \sum_{\substack{(x_i,y_i) \\ \in D_N / D_k}} \lambda(\zeta(x_i), y_i) \right) \\ &= \begin{cases} \underset{D_k \in D_m}{\operatorname{argmin}} \left(\sum_{\substack{(x_i,y_i) \\ \in D_k}} \lambda(\zeta(x_i), y_i) \right), & \text{if m} \geq \text{k} \\ D_m \cup \underset{\substack{D_{k-m} \\ \in D_N / D_m}}{\operatorname{argmin}} \left(\sum_{\substack{(x_i,y_i) \\ \in D_{k-m}}} \lambda(\zeta(x_i), y_i) \right), & \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

Thus, above we want to query instances that will minimize the error. However, since there is indeterminacy in what the missing values will be, we minimize the expectation of ϵ as follows,

$$\begin{aligned} &\underset{D_k \in D_N}{\operatorname{argmin}}\ E[\epsilon] \\ &= \begin{cases} \underset{D_k \in D_m}{\operatorname{argmax}} \left(\sum_{\substack{(x_i,y_i) \\ \in D_k}} P(\zeta(x_i) = y_i) \right), & \text{if m} \geq \text{k} \\ D_m \cup \underset{\substack{D_{k-m} \\ \in D_N / D_m}}{\operatorname{argmin}} \left(\sum_{\substack{(x_i,y_i) \\ \in D_{k-m}}} P(\zeta(x_i) \neq y_i) \right), & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

Equation 13 can be interpreted as, when $m \geq k$ the optimal strategy is to query the k misclassified points that have the highest probability of being correctly classified and when $m < k$, the optimal strategy is to query all misclassified points along with the correctly classified points that have lowest probability of being misclassified. When compared with algorithm 1, it is clear that this is exactly the strategy proposed in JIS, thus proving the optimality of our approach.

3.2 Error Reduction

In this subsection, we derive probabilistic lower bounds on the error reduction of the updated optimal classifier ζ^u_{opt} obtained after training on the updated dataset D^u_N. The updated dataset is created from the values obtained by querying the k instances chosen using our method.

Let ϵ^u_{opt} denote total error of ζ^u_{opt} on D^u_N. If ϵ^u is the total error of ζ on D^u_N, then $\epsilon^u_{opt} \leq \epsilon^u$. This is because ζ^u_{opt} is the optimal classifier on D^u_N, whereas ζ is just one of possible classifiers which the classification algorithm considered during training. Therefore if ϵ was the original error (that is the error of ζ on D_N before querying), then the error reductions of respective classifiers have the following relationship,

$$\begin{aligned} \epsilon^u_{opt} &\leq \epsilon^u \\ \epsilon - \epsilon^u_{opt} &\geq \epsilon - \epsilon^u \\ \Delta\epsilon^u_{opt} &\geq \Delta\epsilon^u \end{aligned} \tag{14}$$

where, $\Delta\epsilon^u_{opt} = \epsilon - \epsilon^u_{opt}$ and $\Delta\epsilon^u = \epsilon - \epsilon^u$. This implies that whenever $\Delta\epsilon^u \geq j$, then $\Delta\epsilon^u_{opt} \geq j$. Thus,

$$P(\Delta\epsilon^u_{opt} \geq j) \geq P(\Delta\epsilon^u \geq j) \tag{15}$$

We now derive exact formulations for $P(\Delta\epsilon^u \geq j)$ as well as more efficient to compute lower bounds when $m \geq k$ and when $m < k$. From equation 15, both of these would be the lower bounds for $P(\Delta\epsilon^u_{opt} \geq j)$, which is our goal.

Let D_p denote a dataset of p instances. Let $L_i^{D_p}$ and $H_i^{D_p}$ denote i instances in the dataset $D_p \subset D_N$ with the lowest and highest scores respectively based on JIS applied to D_N.

If $m \geq k$, then the $P(\Delta\epsilon^u \geq j)$ where $j \in \{1, ..., k\}$, is just the sum of the probabilities of j or more misclassified instances getting correctly classified after querying. Formally,

$$\begin{aligned} P(\Delta\epsilon^u \geq j) &= \sum_{D_j \in D_Q} \left(\prod_{\substack{(x,y_x) \\ \in D_j}} P(\zeta(x) = y_x) \right) \\ &\geq \frac{m!}{(m-j)!j!} \left(\prod_{\substack{(x,y_x) \\ \in L_j^{D_Q}}} P(\zeta(x) = y_x) \right) \end{aligned} \tag{16}$$

where D_Q is the query set as defined in algorithm 1.

The exact formulation in the above equation requires summing over all subsets of j instances in D_m, which may be expensive. However, obtaining the lower bound simply requires multiplying the lowest j scores of the k chosen instances with scaling equal to the number of terms in the summation.

The case where $m < k$ is more involved. In this case, the chosen k instances also contain correctly classified instances and therefore we also have to account for the possibility of these instances getting misclassified upon completion. In

other words, there are more ways than the previous case to get an error reduction of j or more. In particular, we can get an error reduction of $j \in \{1, ..., m\}$ (or more) by correctly classifying j (or more) misclassified instances and correctly classifying the $k-m$ instances that were already correctly classified. We can also get the same error reduction by misclassifying some (say i) previously correctly classified instances, but correctly classifying more than $i+j$ misclassified instances. Thus, to get the formulation for $P(\Delta\epsilon^u \geq j)$ we sum up over the possibilities that (i) all the $k-m$ correctly classified instances are still correctly classified, (ii) all but one of the $k-m$ correctly classified instances are still correctly classified, (iii) all but two of the $k-m$ correctly classified instances are still correctly classified and so on. If $D^{rQ}_{k-m} = D_r \cap D_Q$ denotes the correctly classified instances that we want to query, then $P(\Delta\epsilon^u \geq j)$ is given by,

$$
\begin{aligned}
&P(\Delta\epsilon^u \geq j) \\
&= \prod_{(z,y_z)\in D^{rQ}_{k-m}} P(\zeta(z)=y_z) \left(\sum_{D_j \in D_m} \prod_{(t,y_t)\in D_j} P(\zeta(t)=y_t) \right) \\
&+ \sum_{i=1}^{min(m-j,k-m)} \sum_{D_i \in D^{rQ}_{k-m}} \prod_{(x,y_x)\in D_i} P(\zeta(x)\neq y_x) \\
&\prod_{(z,y_z)\in D^{rQ}_{k-m}/D_i} P(\zeta(z)=y_z) \left(\sum_{D_{i+j}\in D_m} \prod_{(t,y_t)\in D_{i+j}} P(\zeta(t)=y_t) \right) \\
&\geq \prod_{(z,y_z)\in D^{rQ}_{k-m}} P(\zeta(z)=y_z) \frac{m!}{(m-j)!j!} \prod_{(t,y_t)\in L_j^{D_m}} P(\zeta(t)=y_t) \\
&+ \sum_{i=1}^{min(m-j,k-m)} \frac{(k-m)!}{(k-m-i)!i!} \prod_{(x,y_x)\in H_i^{D^{rQ}_{k-m}}} P(\zeta(x)\neq y_x) \\
&\prod_{(z,y_z)\in L^{D^{rQ}_{k-m}}_{k-m-i}} P(\zeta(z)=y_z) \frac{m!}{(m-i-j)!(i+j)!}. \\
&\prod_{(t,y_t)\in L^{D_m}_{i+j}} P(\zeta(t)=y_t) \\
&= \frac{m!}{(m-j)!j!} \prod_{(z,y_z)\in D^{rQ}_{k-m}} \prod_{(t,y_t)\in L_j^{D_m}} P(\zeta(z)=y_z)P(\zeta(t)=y_t) \\
&+ \sum_{i=1}^{min(m-j,k-m)} \frac{(k-m)!m!}{i!(i+j)!(m-i-j)!(k-m-i)!} \\
&\prod_{(x,y_x)\in H_i^{D^{rQ}_{k-m}}} \prod_{(z,y_z)\in L^{D^{rQ}_{k-m}}_{k-m-i}} \prod_{(t,y_t)\in L^{D_m}_{i+j}} P(\zeta(x)\neq y_x) \\
&P(\zeta(z)=y_z)P(\zeta(t)=y_t)
\end{aligned}
\tag{17}
$$

We obtain the lower bound from the above equation in the following manner. The term before the plus sign is lower bounded using the exact same reasoning that we used in equation 16 to lower bound $P(\Delta\epsilon^u \geq j)$ when $m \geq k$. There are two lower bounding substitutions made after the summation over i. The index i indicates the number of correctly classified instances that could be misclassified after updation. Here we first lower bound $\sum_{D_i \in D^{rQ}_{k-m}} \prod_{(x,y_x)\in D_i} P(\zeta(x)\neq y_x) \prod_{(z,y_z)\in D^{rQ}_{k-m}/D_i} P(\zeta(z)=y_z)$.
There are $\frac{(k-m)!}{(k-m-i)!i!}$ terms in this summation and the term with the smallest value is $\prod_{(x,y_x)\in H_i^{D^{rQ}_{k-m}}} P(\zeta(x)\neq y_x) \prod_{(z,y_z)\in L^{D^{rQ}_{k-m}}_{k-m-i}} P(\zeta(z)=y_z)$, which is obtained by considering the misclassification probabilities of i instances that have the highest score in D^{rQ}_{k-m} and the probabilities to classify correctly of $k-m-i$ instances that have the lowest score in D^{rQ}_{k-m}.

The second lower bounding substitution is made for $\sum_{D_{i+j}\in D_m} \prod_{(t,y_t)\in D_{i+j}} P(\zeta(t)=y_t)$ with similar reasoning as that in equation 16. Here we just choose the lowest $i+j$ scores of the m misclassified instances with scaling equal to the number of terms in the summation.

Thus from equations 15, 16 and 17 we have,

$$
\begin{aligned}
&P(\Delta\epsilon^u_{opt} \geq j) \geq \\
&\text{if m} \geq \text{k} \\
&\frac{m!}{(m-j)!j!} \prod_{(x,y_x)\in L_j^{D_m}} P(\zeta(x)=y_x) \\
&\text{otherwise} \\
&\frac{m!}{(m-j)!j!} \prod_{z\in D^{rQ}_{k-m}} \prod_{(t,y_t)\in L_j^{D_m}} P(\zeta(z)=y_z)P(\zeta(t)=y_t) \\
&+ \sum_{i=1}^{min(m-j,k-m)} \frac{(k-m)!m!}{i!(i+j)!(m-i-j)!(k-m-i)!} \\
&\prod_{(x,y_x)\in H_i^{D^{rQ}_{k-m}}} \prod_{(z,y_z)\in L^{D^{rQ}_{k-m}}_{k-m-i}} \prod_{(t,y_t)\in L^{D_m}_{i+j}} P(\zeta(x)\neq y_x) \\
&P(\zeta(z)=y_z)P(\zeta(t)=y_t)
\end{aligned}
$$

These equations provide the lower bounds for error reduction using the proposed approach. For particular applications, the above equations can be computed to quickly get a feel of how effective our querying strategy is likely to be and evaluate if they meet the application needs. If similar bounds can be computed for other querying methods, one can compare them and choose the appropriate method without empirical studies.

4. INDEPENDENT INSTANCE SELECTION (IIS)

In this section, we first discuss a simplified univariate version of our method, where we analyze each missing feature for an instance independent of the other missing features. We then show that restricting this method to only categorical features makes it equivalent to the EU method of [13].

4.1 Method Description

In algorithm 1, we compute a function $f(.)$ over all the missing features. However, it is possible to simplify the approach and compute $f(.)$ for each missing feature of an

Algorithm 2 Changes to algorithm 1 for IIS during function and score computation. The other steps are identical. The below steps are within the for loops which go over all $x \in D_m$ and over all $x \in D_r$ in algorithm 1.

$score_x = 0$
for all $A \in M(x)$ **do**
  Compute $f(x^A) \in S_x^A$ where $x^A \in V_x^A$ such that, $\forall$ $v \in V_x^A$, $\zeta(x^{(A=v)}) = y_x\{x^{(A=v)}$ denotes an instance created by filling in the missing value of feature A in x with $v.\}$
  $score_x = score_x + P(f(x^A) \in S_x^A)$
end for
$Q = Q \cup score_x$

instance independently. This could be useful if there is insufficient data to accurately estimate the probabilities of the functions over multiple missing features.

In this case the function $f(.)$ would be computed similarly as before except that we would compute it for each missing feature assuming the already filled in values for the other missing features for that instance. Thus, for an instance we would have as many functions as there are missing features. The criteria satisfied by each of these functions would correspond to sufficient conditions to correctly classify the instance given everything else. Hence, the score for an instance in this case would be the sum of the probabilities of each of those functions satisfying their respective critera. These updates to algorithm 1 that give us the IIS method are shown in algorithm 2.

4.2 Relation to Expected Utility Method

In the EU method, given a dataset with categorical variables we find the expected utility for each missing entry (instance-feature value). For example, if $A \in M(x)$ is a missing feature for the instance (x, y_x) with possible categorical values $V^A = \{a_1, a_2, ..., a_g\}$ and $x^{(A=a_j)}$ denotes an instance created by filling in the missing value of A in x with a_j, then the utility function $\mathcal{U}(.)$ is defined as:

$$\mathcal{U}(x^A = a_j) = \lambda(\zeta(x), y_x) - \lambda(\zeta(x^{(A=a_j)}), y_x) \tag{18}$$

where $\lambda(.,.)$ is a $0-1$ classification loss function. Based on this definition of utility, the expected utility for querying entry x^A is given by:

$$E[\mathcal{U}(x^A)] = \sum_{j=1}^{g} P(A = a_j)\mathcal{U}(A = a_j) \tag{19}$$

Consequently, the score for the instance x can be defined as the sum of the expected utilities of the corresponding missing features,

$$\text{score}_x = \sum_{A \in M(x)} E[\mathcal{U}(x^A)] \tag{20}$$

From equation 18 we can see that the utility function can only have values from the set $\{1, 0, -1\}$. If a misclassified instance gets correctly classified after substitution, then we obtain a 1. If there is no change in terms of being correctly or incorrectly classified, then we get a 0. Whereas if a correctly classified instance gets misclassified after substitution we get a -1. Thus, in the case that an instance is initially misclassified the value of the utility function for different combinations of missing features and corresponding substituted value can only be 1 or 0. This is because for a misclassified instance $\lambda(\zeta(x), y_x) = 1$ and $\lambda(\zeta(x^{(A=a_j)}), y_x) \in \{0, 1\}$. Thus, the utility function is 1 only for those substitutions that result in correctly classifying the instance, else it is 0.

Given this, the expected value of this utility function computed according to equation 19 would signify the probability of correct classification if we were to query for the missing value. The final score would be the sum of these probabilities as follows,

$$\text{score}_x = \sum_{A \in M(x)} \sum_{\substack{a \in V^A | \zeta(x) \neq y_x, \\ \zeta(x^{(A=a)}) = y_x}} P(A = a) \tag{21}$$

Similarly, if an instance is correctly classified, then the utility function is -1 only for those substitutions that result in misclassifying the instance, else it is 0. Given this, the expected value of this utility function computed according to equation 19 would be the negative of the probability of misclassification if we were to query for the missing value. The final score again would be the sum of these probabilities as follows,

$$\text{score}_x = -\sum_{A \in M(x))} \sum_{\substack{a \in V^A | \zeta(x) = y_x, \\ \zeta(x^{(A=a)}) \neq y_x}} P(A = a) \tag{22}$$

On sorting these scores, we observe that the initially misclassified instances would always have scores no less than the correctly classified instances because from equations 21 and 22 we see that the scores for the misclassified instances are lower bounded by 0, while the scores for the correctly classified instances are upper bounded by 0. This means that if we were to choose k instances to query and $m \geq k$, we would choose from the misclassified instances that have the highest chance of being correctly classified based on the sum of the marginals. If $m < k$, then we would choose all the misclassified instances followed by choosing $k - m$ correctly classified instances that have the least chance of being misclassified. This is precisely what we are doing in IIS. This shows that if we restrict IIS to only categorical features it is equivalent to the EU method.

5. EXPERIMENTS

Using the techniques proposed in this paper, we next empirically validated them on three UCI datasets – *credit approval*, *adult*, and *spambase* and two e-commerce datasets – *priceline*, and *etoys* which were used in previous works [14]. We also performed these experiments with varying levels of noise to test the robustness as compared to its competitors. We first describe the setup used in all the experiments followed by the experimental results and their corresponding observations.

5.1 Setup

We compared our JIS technique with its univariate version IIS and with the most sophisticated methods used for this problem in literature namely; Error Sampling and Expected Utility approaches. Since EU requires categorical inputs, in datasets which contain continuous attributes, we

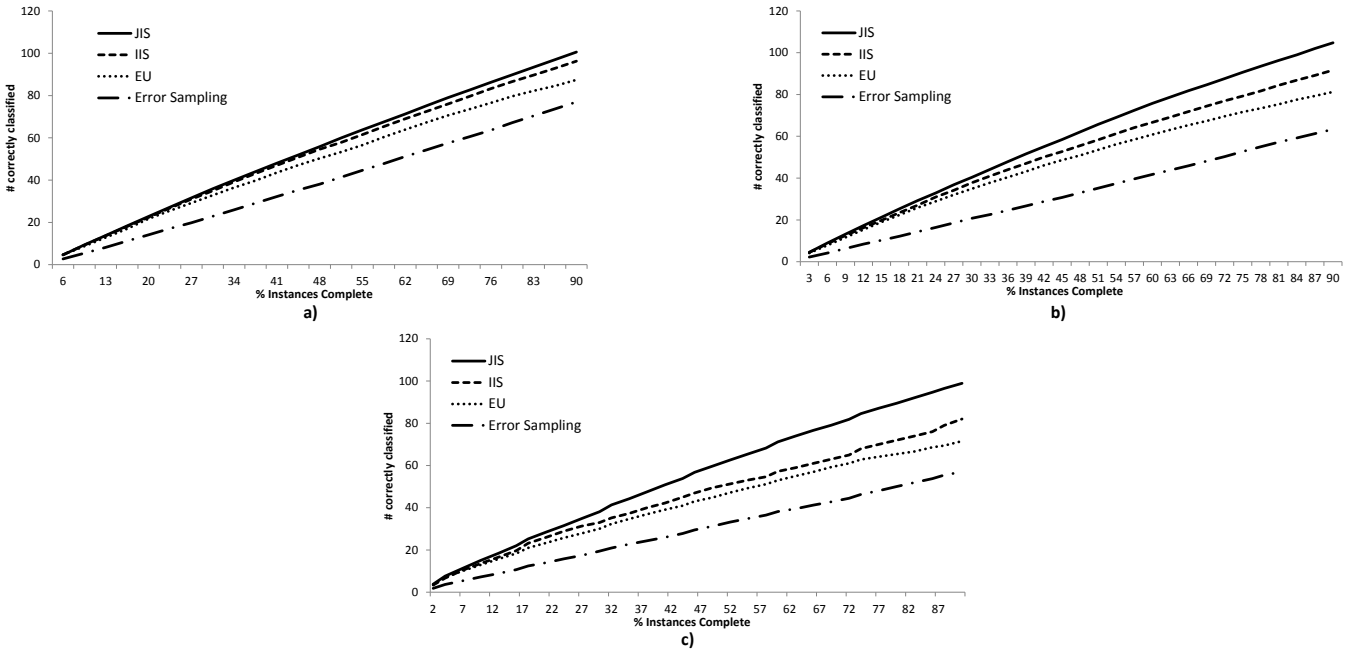

Figure 1: Above we see the performance of the four methods on the credit approval dataset. a) is the case with no noise, b) is the case with 20% noise and c) is the case with 40% noise. For clarity we do not show the confidence intervals here but they are provided in table 1 along with results for other datasets.

first discretize the continuous attributes into 10 equal bins calculated from their range of values and then apply EU.

We performed 10 runs of 10 fold cross validation which is the same experimental methodology as [13], for each of the settings described next. For each instance in a dataset, we randomly set 50% of its input features to be missing. To test robustness of the different methods we introduced label noise, where we randomly changed the labels of η percent of the instances. We ran experiments on each dataset for three values of η, which are 0% (no noise), 20% and 40%. For each noise level we incrementally completed instances in steps of 10 by querying them based on the decisions of the respective methods with the model being updated after each completion. We report the total number of queried instances correctly classified up until each completion averaged across all runs (with a 90% confidence interval) as the measure of performance. We used SVM with RBF kernel (Weka's implementation) in all our experiments. The density estimation for JIS and IIS was done using the well known kernel density estimation technique with gaussian kernels.

5.2 Observations

Figures 1(a), 1(b) and 1(c) show the performance of the four methods on the credit approval dataset at increasing levels of noise. We observe that JIS is consistently superior to all the other methods. In fact, the difference in performance becomes especially noticeable as the noise levels increase. From table 1 we see that a similar phenomenon occurs on all the other datasets as well, demonstrating that JIS is significantly more robust than its competitors. It also implies that as the classification problem gets harder, choosing which instances to complete intelligently based on joint information (as we do in JIS) becomes more critical, than simply picking misclassified instances randomly and correctly classified instances based on classification uncertainty (as is the case with Error Sampling), or picking instances based on marginals (as is the case with both EU and IIS).

Though IIS is better than EU on datasets with continuous attributes, the improvement is not as much as that obtained with JIS. This is not surprising since both IIS and EU are based on marginals and are equivalent when the data is categorical as shown before. The improved performance of IIS can be attributed to the fact that no discretization of continuous attributes is required for IIS, which is a necessary approximation for the applicability of EU.

All three methods JIS, IIS and EU are better than ES, which substantiates our intuition of picking instances to query based directly on their likelihood of being correctly classified instead of relying on other measures.

6. DISCUSSION

In this paper we described a novel technique for instance completion for improving classification performance on datasets containing missing input information. We proved that this technique is optimal given the currently available information. We also proved probabilistic lower bounds on the error reduction of the optimal classifier learned over the updated dataset that is obtained after following our querying procedure. We described a simplified univariate version of our method, which when restricted to categorical features was equivalent to the expected utility method. We then empirically demonstrated that our method outperforms other competing methods over a wide range of completion percentages and different noise levels.

Previous works have also suggested incorporating a cost matrix $\mathcal{C}$ which indicates the cost of querying each entry. However, empirical studies in these works were performed on

	No Noise				Noise Level 20%				Noise Level 40%			
% complete	10	20	50	90	10	20	50	90	10	20	50	90
Credit Approval												
JIS	**9.4 +-0.8**	**22.8 +-1.5**	**59.9 +-2.8**	**100.6 +-4.8**	**17.3 +-0.9**	**29.2 +-1.3**	**58.4 +-2.4**	**98.9 +-4.8**	**15.1 +-0.5**	**28.5 +-0.8**	**59.7 +-2.3**	**96.5 +-4.8**
IIS	9.2 +-1	22.4 +-2	57.8 +-3.1	96.3 +-5.1	16.3 +-1.2	27.06 +-2.2	52.7 +-3.4	86.8 +-5.5	13.6 +-0.7	26 +-1.4	49.5 +-3.6	79 +-5.6
EU	8.7 +-0.9	21.7 +-2.2	53.3 +-3.2	87.4 +-5.1	15.4 +-0.7	25.8 +-1.8	48.7 +-4.6	77.6 +-5	12.9 +-0.8	23.4 +-1.6	44.9 +-3.8	69.4 +-5.3
ES	5.4 +-1.6	14.1 +-2.3	41.4 +-2.4	76.9 +-3.3	8.5 +-1.2	14.3 +-1.7	30.8 +-2.4	59.2 +-3.3	7.3 +-0.7	14 +-1	31.3 +-1.3	55.5 +-1.7
Adult												
JIS	**8.1 +-0.4**	**15 +-0.8**	**32.4 +-2.6**	**56.4 +-3.9**	7.1 +-0.7	14.9 +-1.1	**35.9 +-1.8**	**58.1 +-2.3**	**8.6 +-0.7**	**16.1 +-1**	**36.3 +-1.8**	**59.8 +-2.8**
IIS	7.8 +-0.6	14.8 +-0.9	31.5 +-2.2	45.9 +-3.4	**8 +-0.4**	**15.3 +-1.1**	33.1 +-2	54.4 +-3	7.5 +-0.6	15 +-0.8	32.2 +-1.7	47.7 +-3.9
EU	7.4 +-0.7	14.6 +-1	30.1 +-1.7	44.6 +-2.2	7.8 +-0.4	14.3 +-1	30.7 +-1.9	49.4 +-2.9	7.1 +-0.5	13.2 +-0.8	29.6 +-1.6	41.8 +-3.6
ES	4.1 +-0.5	8.2 +-0.9	21.1 +-1.3	41.5 +-2.2	3.8 +-0.7	7.5 +-1	19.6 +-1.9	36.2 +-2.4	3.9 +-0.7	7.1 +-0.7	18.4 +-1.5	32.9 +-1.5
Spambase												
JIS	**8.7 +-0.4**	**15.6 +-0.9**	**33.8 +-1.4**	**61.7 +-1.5**	**4.9 +-0.5**	**10.1 +-0.5**	**29 +-1.6**	**59.4 +-2**	**4.7 +-0.5**	**10 +-0.4**	**25 +-1.8**	**58.1 +-2**
IIS	8.2 +-0.4	15.1 +-0.5	32.7 +-0.8	60.2 +-1	4.8 +-0.5	9.6 +-0.7	27.4 +-1.3	56.7 +-1.8	5.8 +-0.6	11.6 +-1	24.3 +-1.2	56.4 +-1.8
EU	7.2 +-0.2	13.5 +-0.5	29.3 +-0.6	56.7 +-1	4 +-0.5	8.24 +-0.7	20.2 +-1.3	42.7 +-1.8	3.8 +-0.5	7.7 +-0.8	20.8 +-1.2	42.1 +-1.5
ES	5.3 +-0.5	11.1 +-0.7	28.4 +-1.1	52.5 +-1.8	2 +-0.6	4.5 +-0.9	15.9 +-1.6	34.8 +-2.5	2.2 +-0.6	4.5 +-0.9	11.9 +-1.6	33.8 +-2.5
Priceline												
JIS	**8 +-0.7**	**15.7 +-0.9**	**37.7 +-1.5**	**65.7 +-2.3**	**7.2 +-0.5**	**14.3 +-1.1**	**36.8 +-2.3**	**67.6 +-3.3**	**7.6 +-0.6**	**14.6 +-1.3**	**37.8 +-2.3**	**69.2 +-3.2**
IIS	7.5 +-0.6	15.3 +-1	37.1 +-1.5	63.3 +-2	6.8 +-0.4	14.2 +-0.6	35.5 +-0.8	60.7 +-1.9	6.8 +-0.8	13.6 +-1.3	32.9 +-1.1	57.4 +-2.3
EU	6.9 +-0.6	14.1 +-0.9	34.7 +-1.4	58.1 +-1.8	6.5 +-0.4	13.6 +-0.6	32.8 +-0.8	56 +-1.7	6.3 +-1	12.6 +-1.2	29.4 +-1	52 +-2.1
ES	6.3 +-0.5	12.3 +-0.9	31.5 +-1.4	57.7 +-1.8	3.5 +-0.6	8.5 +-0.7	23.3 +-2.2	42 +-3.3	3.8 +-0.5	7.6 +-0.6	19.5 +-1	37.7 +-1
Etoys												
JIS	**5.6 +-0.9**	**9.2 +-0.9**	**21.4 +-1.2**	**40.8 +-1.2**	**2.5 +-0.3**	**10.2 +-0.8**	**17.8 +-1.2**	**45.5 +-0.8**	**3.5 +-0.3**	**6.38 +-0.3**	**20.71 +-1.4**	**48.5 +-1.9**
IIS	3.2 +-0.8	6.5 +-1	18.1 +-1	37.9 +-1	2 +-0.4	8 +-1	16.5 +-1.3	43.5 +-1.5	1.7 +-0.8	3.8 +-1	18.5 +-1.5	41.5 +-1.8
EU	2.8 +-0.7	5.9 +-0.9	16.5 +-0.8	34.7 +-1	1.8 +-0.4	7.4 +-0.9	13.6 +-1.1	35.8 +-1.3	1.6 +-0.7	3.5 +-0.9	16.5 +-1.3	37.1 +-1.5
ES	2 +-0.5	4.1 +-0.9	14.5 +-1.2	33.5 +-1.2	1.6 +-0.4	4.1 +-0.5	12.35 +-1	33.5 +-0.9	2.91 +-0.6	4.8 +-0.9	15.7 +-1.8	35.5 +-1.9

Table 1: The above table shows the performance of the different methods with increasing noise levels and increasing percentage of complete instances. The entries in bold indicate the best performance of the four methods in the specific scenarios.

real datasets where this matrix was not available. Obtaining such a cost matrix in a practical scenario is quite challenging and we therefore did not include it in our problem statement so as to make the exposition clearer. Nevertheless, both the proposed methods JIS and IIS can in fact be easily extended to scenarios where this additional information is available.

If c_i^j is the cost of querying the attribute j of instance x_i, then the total cost for querying all the missing entries for an instance x_i is given by, $C_i = \sum_{j \in M(x_i)} c_i^j$. With this, the score for each instance x_i obtained from JIS and IIS would just be scaled down by C_i. Thus, if $score_{x_i}$ was the original score for x_i, the new score would be given by,

$$score_{x_i}^{new} = \frac{1}{C_i} score_{x_i} \tag{23}$$

For the IIS method if we were just trying to find which entries to query, we would scale each marginal probability down by the corresponding cost. However, since we are interested in instance completion, where upon selecting an instance we query all of its missing features, it is appropriate to scale the score for the instance down by the total instance level cost. This also applies to the EU method, which again would be equivalent to the IIS method restricted to categorical features.

It is also useful to note that JIS and IIS techniques can be viewed as two ends of a spectrum where, while the JIS method computes a function over all missing features for an instance, the IIS method computes functions for each of them in isolation. Indeed, we can develop an entire spectrum of methods which compute functions over different sized subsets of missing features and then compute probabilties over these functions whose sum then leads to the final score. Such methods could be of interest as different methods in this spectrum might perform superior to the others for different percentages of missing values. The challenge here however, is to efficiently identify the best subsets. This would require further investigation.

From a theoretical standpoint, it would be interesting to study how our querying method compares to a method that would result in the best updated classifier. Note that our method is optimal given the current classifier but it does not guarantee that the classifier trained on the corresponding updated dataset is the best among all possible classifiers that would be produced through other querying mechanisms. In other words, there might be other querying strategies that are better than ours from the point of view of the final updated classifier. Even if it turns out that our method isn't optimal from this other perspective, it would be interesting to see how much worse we are (i.e. find the regret). This would probably require assuming some amount of smoothness in the learning of the classification algorithm with respect to the change in inputs. An alternative strategy would be to restrict the hypothesis space to linear or some other class of functions. While these directions have been of some interest in the area of traditional active learning [3], to the best of our knowledge, there hasn't been any such work related to our problem of instance completion.

Acknowledgement

We would like to thank Maytaal Saar-Tsechansky and Prem Melville for providing the e-commerce datasets. We would also like to thank the reviewers and the meta-reviewer for their constructive comments.

7. REFERENCES

[1] M. Bilgic and L. Getoor. Voila: Efficient feature-value acquisition for classification. In *AAAI '07: Proceedings of the 22nd National Conference on Artificial Intelligence*, 2007.

[2] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2):201–221, 1994.

[3] S. Dasgupta and J. Langford. Active learning tutorial. In *Proceedings of the 26th International Conference on Machine Learning*, ICML '09, 2009.

[4] P. Frazier. *Knowledge-gradient Methods for Statistical Learning*. Princeton Univ. Press, 2009.

[5] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford, 3 edition, 2001.

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

[7] P. Kanani and P. Melville. Prediction-time active feature-value acquisition for customer targeting. In *Proceedings of the Workshop on Cost Sensitive Learning, NIPS 2008*, 2008.

[8] A. Kapoor and R. Greiner. Learning and classifying under hard budgets. In *In Proceedings of the European Conference on Machine Learning (ECML-05*, pages 170–181. Springer, 2005.

[9] D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1994.

[10] D. J. Lizotte and O. Madani. Budgeted learning of naive-bayes classifiers. In *In Proceedings of 19th Conference on Uncertainty in Artificial Intelligence (UAI-2003*, pages 378–385. Morgan Kaufmann, 2003.

[11] P. McCullagh and J. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall, 1990.

[12] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pages 483–486. IEEE Computer Society, 2004.

[13] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. An expected utility approach to active feature-value acquisition. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '05. IEEE Computer Society, 2005.

[14] M. Saar-Tsechansky, P. Melville, and F. Provost. Active feature-value acquisition. *Manage. Sci.*, 55(4):664–684, 2009.

[15] M. Saar-tsechansky and F. Provost. Active sampling for class probability estimation and ranking. In *Machine Learning*, pages 153–178, 2004.

[16] B. Settles. Active learning literature survey. Technical report, 2010.

[17] V. S. Sheng and C. X. Ling. Feature value acquisition in testing: A sequential batch test algorithm. In *In Proceedings of 2006 International Conference on Machine Learning (ICML 2006*, pages 809–816, 2006.

[18] V. Vapnik. *Statistical Learning Theory*. Wiley & Sons, 1998.

[19] Z. Zheng and B. Padmanabhan. On active learning for data acquisition. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '02. IEEE Computer Society, 2002.