Mach Learn (2015) 100:425–447
DOI 10.1007/s10994-015-5500-5

Improving classification performance through selective instance completion

Amit Dhurandhar[1] · **Karthik Sankaranarayanan**[2]

Received: 9 November 2014 / Accepted: 20 April 2015 / Published online: 14 July 2015

Abstract In multiple domains, actively acquiring missing input information at a reasonable cost in order to improve our understanding of the input–output relationships is of increasing importance. This problem has gained prominence in healthcare, public policy making, education, and in the targeted advertising industry which tries to best match people to products. In this paper we tackle an important variant of this problem: Instance completion, where we want to choose the best k incomplete instances to query from a much larger universe of N ($\gg k$) incomplete instances so as to learn the most accurate classifier. We propose a principled framework which motivates a generally applicable yet efficient meta-technique for choosing k such instances. Since we cannot know a priori the classifier that will result from the completed dataset, i.e. the final classifier, our method chooses the k instances based on a derived upper bound on the expectation of the distance between the next classifier and the final classifier. We additionally derive a sufficient condition for these two solutions to match. We then empirically evaluate the performance of our method relative to the state-of-the-art methods on four UCI datasets as well as three proprietary e-commerce datasets used in previous studies. In these experiments, we also demonstrate how close we are likely to be to the optimal solution, by quantifying the extent to which our sufficient condition is satisfied. Lastly, we show that our method is easily extensible to the setting where we have a non-uniform cost associated with acquiring the missing information.

Keywords Instance completion · Active feature acquisition · Missing data

Editors: João Gama, Indrė-Žliobaitė, Alípio M. Jorge, and Concha Bielza.

✉ Amit Dhurandhar
sadhamanus@gmail.com; adhuran@us.ibm.com

Karthik Sankaranarayanan
kartsank@in.ibm.com

[1] IBM T.J. Watson, Yorktown Heights, NY, USA

[2] IBM India Research Lab, Bangalore, India

1 Introduction

Incomplete or missing data is a bane that plagues insightful data analysis in almost every industry. Researchers and practitioners alike have come up with a variety of solutions to deal with this issue. On the one hand, there are statistical or machine learning techniques (Han and Kamber 2011) that try to best estimate the missing information based on known data. On the other hand, there are domain specific data filling techniques (Weiss et al. 2013) that use expert knowledge to estimate the missing information. These two strategies have been the most popular in literature when it comes to dealing with missing data. In some cases there is a third and potentially better alternative, which involves trying to actively acquire the missing information within certain cost constraints. In healthcare for example, to get a better understanding of the efficacy of a flu vaccine in relation to demographics, private and government health agencies might want to contact healthy and sick individuals who were administered the vaccine and query them about their lifestyle/habits. In education, to evaluate standardized tests, education boards and schools might want to obtain more information about their students to find correlations of which students performance well and their backgrounds. Of course in any practical scenario it is virtually impossible to ask everyone and acquire all the information. The question thus becomes, which individuals must we target to learn the most about these associations.

This problem is also seen in the targeted advertising industry, which tries to best match people to products. Hence, while you might be able to estimate correlations or more generally build classification models by associating available demographic features to consumer buying decisions, such models are generally far from perfect. For example, we may not know the age and/or gender and/or geography of some customers, which might be critical in determining their preference for a certain product. The question then again becomes, which individuals must we target to maximize the improvement in prediction/classification performance.

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 with incomplete input information but with known outputs, which instances when queried would maximize the classification performance?* This problem was introduced by Melville et al. (2004) and is commonly referred to as *Instance completion*. In the application domains described before, this formulation would correspond to choosing k people to call that will most likely significantly enhance classification accuracy having obtained their additional information. The better the classifier the more confidence we are likely to have in the insight it provides by identifying features that it deems significant. This is complementary to the traditional *active learning* problem where only a single attribute (i.e. the label) is missing. It can, however, be considerably harder to design effective techniques for *instance completion*, since there could be multiple different sets of (input) attributes missing for each of the instances.

Other than this being a highly relevant problem in todays world, one of the primary motivations for this work is that there is still a significant performance gap between the state-of-the-art methods (Sankaranarayanan and Dhurandhar 2013; Melville et al. 2005; Saar-Tsechansky et al. 2009; Kanani and Melville 2008) and the best possible solution. This is seen in Fig. 1, where we see the accuracy of various methods being much worse than the best possible solution at different completion percentages. At every iteration/completion percentage, the *Best Case* solution is found by exhaustively searching across all possible k incomplete instances, and then choosing those k instances which upon completion and retraining of the classifier, lead to the highest accuracy over a (10 %) validation set that was

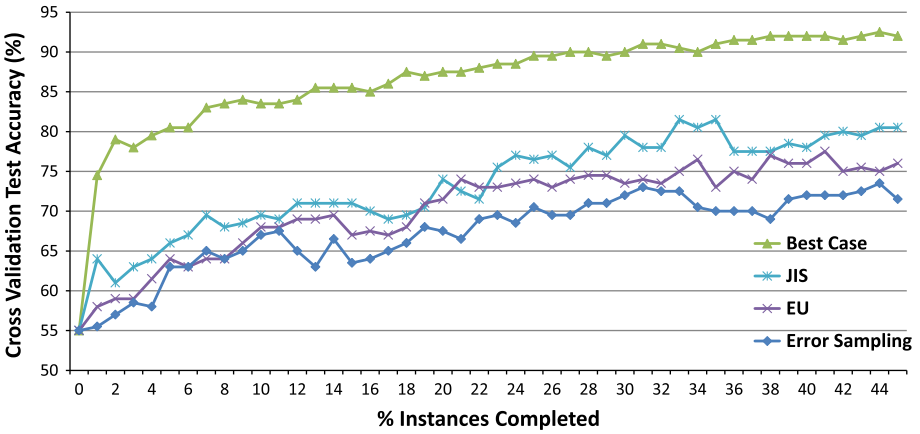

Fig. 1 The 10-fold test set accuracy of the state-of-the-art methods relative to the best possible set of $k = 2$ instances (found exhaustively) that could be queried at each iteration for the UCI credit approval dataset. We incrementally query sets of k instances based on the different strategies and at each stage retrain a SVM with radial basis kernel (Color figure online)

part of the training.[1] These results are on the Credit Approval UCI dataset where a random 50 % of the attributes are assumed to be missing for each incomplete instance (in keeping with the experimental methodology used in previous works (Sankaranarayanan and Dhurandhar 2013; Melville et al. 2005).[2] A qualitatively similar performance gap is also seen on other datasets from previous studies.

We believe that an important conceptual reason why such a performance gap still exists is that none of the existing methods attempt to *explicitly* approach the best possible classifier. State-of-the-art methods such as joint instance completion (JIS) (Sankaranarayanan and Dhurandhar 2013) and expected utility (EU) (Melville et al. 2005) are optimal at the instance and entry (i.e. instance-feature combination) level respectively but only with respect to the *current* classifier. In other words, there could be other querying mechanisms that would lead to better classifiers that are closer to the final classifier, than those obtained by using the above methods, where the final classifier is the one obtained by training on the complete dataset.[3]

In this paper, we thus provide a general framework built on the philosophy of trying to get as close as possible to the final classifier as we query instances in sets of k and update the classifier at each iteration. In particular, we make the following contributions,

- We define a novel, intuitive distance metric between two classifiers relative to a dataset that is independent of the classification hypothesis/function classes, but rather dependent on their performance.
- We show equivalance between the original problem of minimizing the distance between the next classifier and the final classifier, and the problem of maximizing the error reduction between successive classifiers (obtained by querying instances in sets of k), and then

[1] We chose $k = 2$ since exhaustively searching across larger values of k becomes computationally too intensive to obtain the best-case results.

[2] Thus, each incomplete instance can have different sets of attributes missing.

[3] We assume that the classifier trained on the completed dataset (the final classifier), though unknown, will be the best possible one and therefore, we want to learn it as early as possible, which we feel is a reasonable goal.

upper bounding it by a function that is independent of the final classifier. We also provide an exponential (upper) bound on the error reduction, which reaffirms our strategy.
- We provide a sufficient condition under which optimizing this upper bound would lead to the same solution as optimizing the original function.
- Motivated by these developments, we provide a meta-algorithm to choose k instances. To illustrate its easy of applicability to popular classification algorithms we instantiate it for linear and non-linear SVMs as well as logistic regression.
- We demonstrate the superiority in performance of our method over the state-of-the-art on a collection of UCI and proprietary e-commerce datasets. In these experiments, we also show how close we are likely to be to the optimal solution, by presenting the extent to which the derived sufficient condition is satisfied.

The rest of the paper is organized as follows: In Sect. 2, we position our work relative to advances in traditional active learning as well as relative to our problem of instance completion. In Sect. 3, we provide a general overview of our method and describe the interplay between the following three sections. In Sect. 4, we describe in detail our framework. This includes defining the performance based distance function and the sufficient condition for optimality. In Sect. 5, we elucidate our method that is motivated by the developments in the previous section. In Sect. 6, we provide example instantiations of our technique to linear and non-linear SVMs as well as logistic regression. This section showcases how the univariate probabilities that our method in the previous section requires can be computed. In Sect. 7, we empirically compare our method with other state-of-the-art methods on real data and report how close we are to satisfying the derived sufficient condition. In Sect. 8, we discuss how our method can be extended given a cost matrix along with promising future directions.

2 Related work

The problem setting described in this paper is quite different from traditional active learning (Settles 2010; Cohn et al. 1994), which involves the complementary problem of missing outputs with all inputs known and the goal being to query those output labels that can maximize classification performance. Other related work includes work on budgeted learning (Lizotte and Madani 2003; Kapoor and Greiner 2005), where one is trying to find which entry (i.e. instance-feature combination) to query given a certain cost constraint. Knowledge gradient methods (Frazier 2009) are a different class of methods which also try to find the optimal entry to query that will maximize the gain in information. However, both these classes of methods are suited for a setting that is different from ours. Budgeted learning and knowledge gradient methods are generally modeled as Markov decision processes (MDPs) and, as mentioned before, try 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[4] for the corresponding MDP to choose from, which would make this framework computationally infeasible.

While the above methods are designed for selecting one entry at a time, there are other methods in traditional active learning that try to optimally select a set of entries at a time and are referred to as batch mode active learning methods (Hoi et al. 2006; Wang and Ye

[4] ! denotes factorial.

2013; Guo and Schuurmans 2007). Loosely speaking, these methods try to minimize the overlap of information in the chosen set along with classification uncertainty. However, none of them can be easily adapted to our setting as we can have an arbitrary number of attributes missing for any particular instance. Moreover, many of them are based on heuristics such as minimizing the Fisher information or uncertainty of classification, which does not necessarily translate into obtaining the best possible classifier.

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) (Kanani and Melville 2008; Bilgic and Getoor 2007, 2011; Sheng and Ling 2006). The authors in one of these works (Kanani and Melville 2008) try to query instances in the test set so as to minimize classification uncertainty, while in other works (Bilgic and Getoor 2007, 2011; Sheng and Ling 2006) the goal is to query test instances so as to select the most important features. There are also works (Desjardins et al. 2010) that assume that a certain set of features are known for all instances, with the remaining features missing, and given feature acquisition costs, the goal is to decipher which features should be queried for which instances. While in works (Moon et al. 2014), both labels and features may be missing for an instance and the goal is to find out which of the two should be queried. Their method uses Expectation Maximization to impute the missing features for each instance, which can be expensive in real settings where hundreds to thousands of features could potentially be missing. Clearly, all these lines of work address a goal different from ours.

There are works (Sankaranarayanan and Dhurandhar 2013; Zheng and Padmanabhan 2002; Melville et al. 2004, 2005) that address the same problem as ours. In one of these works (Zheng and Padmanabhan 2002), 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 another of these works (Melville et al. 2004), a method called Error Sampling (ES) is described for picking 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 (Lewis and Catlett 1994; Saar-Tsechansky and Provost 2004). 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 Melville et al. (2005), the authors provide a method to score each missing entry based on the calculation of expected utility. 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 obtaining a score. The most recent work (Sankaranarayanan and Dhurandhar 2013) chooses (misclassified) instances that have the highest probability of correct classification w.r.t. the current classifier. One of the key improvements in this method over and above previous methods is that it jointly estimates the probability of correct classification, though requiring only univariate probability estimation.

One of the conceptual deficiencies of the above approaches is that they do not score instances based on their likelihood of producing the best possible classifier that would be *as close as possible* to the final classifier. Even the EU and JIS methods, which are optimal at the entry and instance level respectively, are only so w.r.t. the current classifier, not the final one. Addressing this deficiency is the primary motivation behind our work. Moreover, the

probability estimations involved in the resultant approach are still just univariate keeping the implementation tractable.

3 Overview

The crux of our method is described in Sect. 5. However, Sects. 4 and 6 help in providing a more complete picture in terms of the motivation behind our approach and an illustration of its inner workings respectively.

In Sect. 4, we define a distance function between classifiers, based on their performance on a given dataset and independent of their functional form. This makes the distance easily computable. We propose the problem, where we want to minimize the distance between the next classifier that we build and the final classifier that we would ideally have if the dataset was complete. Using properties of our distance function we relax this problem into the problem where we want to query instances that maximize the distance between the current and the next classifier. Given that our goal is to provide a generally applicable meta-algorithm, in Sect. 5 we provide such an (heuristic) algorithm based on asymmetric uncertainty, which is motivated by developments in the previous section. The algorithm requires computation of multivariate probabilities over the missing features of an instance, which could be highly inefficient to compute. However, we provide a (approximate) strategy, which requires only univariate density estimation and is quite effective as is witnessed in the experimental section. Instantiation of this strategy to popular classification techniques such as linear and non-linear SVMs, and logistic regression is given in Sect. 6.

Hence, the next three sections together provide a well rounded view of our proposed solution. In a certain sense we move from more conceptual to operational as we proceed through these sections.

4 Framework

Let $X \times Y$ denote the input–output space and let $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be a subset of this space be a dataset. Let $\zeta_i : X \to Y$ denote the classifier built on the complete instances of S available after the ith iteration of querying incomplete instances in batches of size k, where $i \in \{0, \ldots, f\}$. ζ_0 denotes the initial classifier. After querying f times the dataset is complete so that ζ_f denotes the final classifier.

We next develop a distance function between classifiers relative to the dataset so that such a metric is agnostic to the intricacies of the functional representations or hypothesis classes of classifiers. Let $\lambda(.,.,.)$ be a loss function such that for any real a, b, c,

$$\lambda(a, b, c) = \begin{cases} 1, & \text{if } b = c \text{ and } a \neq c \\ 1, & \text{if } a = c \text{ and } b \neq c \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Using this $\lambda(.,.,.)$, we define the distance between classifiers ζ_i and ζ_j on the dataset S as,

$$d_S(\zeta_i, \zeta_j) = \sum_{(x,y)\in S} \lambda(\zeta_i(x), \zeta_j(x), y) \tag{2}$$

Put simply, the distance between any two classifiers relative to a dataset is the number of instances in the dataset that exactly one of them correctly classifies. In the binary label

setting, this corresponds to the number of instances that they classify differently. In the multilabel setting, if the two classifiers classify an instance in different classes both of which are incorrect, then the distance on that instance is still zero. This instance adds to the distance only when exactly one of them correctly classifies it. It is easy to see that such a metric that does not depend on the function classes of classifiers is particularly useful when we consider non-linear classifiers (viz. SVMs with RBF kernel, etc.).

In traditional active learning, there has been some work on trying to reach the final classifier under restricted settings (Dasgupta and Langford 2009). In particular, the work on the disagreement coefficient (Hanneke 2007) is the most relevant as their distance function is essentially an expectation of a simplified version of our metric which just counts the number of instances the classifiers label differently. However, as is the case with the works mentioned before, it is not straightforward to extend their work and obtain a practical generally applicable algorithm for our instance completion setting. In fact, even in the traditional active learning setting, the work presented is applicable primarily for binary labeling and simple hypothesis classes in low dimensions. In higher dimensions, additional distributional assumptions have to be made. Moreover, as opposed to using disagreement between classifiers as a distance function, our distance function differentiates classifiers based on their performance even in the multiclass setting, which is desirable. This is the case since, as mentioned before, the distance between any two classifiers is zero even if they classify instances in a dataset into different classes but all of which are incorrect. These classifiers are thus considered to be equivalent, which is not the case when using disagreement as a distance. A consequence of this, as we will soon see, is that our distance function yields a more efficient strategy, that does not have to consider all pairs of labels to maximize the distance between successive classifiers in each iteration.

We now state certain properties of our distance function which are useful in the developments that follow.[5]

Proposition 1 *$d_S(.,.)$ is non-negative, symmetric and satisfies triangle inequality.*[6]

Now at any iteration of querying i let $\mathcal{I}_{i-1}$ and C_{i-1} denote the set of incomplete and complete instances before querying respectively. From $\mathcal{I}_{i-1}$ we want to query k incomplete instances q^k such that,

$$Q_i^k = \underset{q^k | q^k \in \mathcal{I}_{i-1}}{\operatorname{argmin}} \; d_S(\zeta_i, \zeta_f) \tag{3}$$

Notice here that we want to reach the final classifier as fast as possible, since we assume that ζ_f is the best possible classifier i.e. $\forall i \in \{0, \ldots, f-1\}, d_S(Y, \zeta_f) \leq d_S(Y, \zeta_i)$, where Y are the labels of the instances in S. In other words, ζ_f has the lowest error relative to the other classifiers that can be learned with a chosen classification algorithm on an incomplete S.

Even though in Eq. 3, ζ_f is not known, we do know that $d_S(.,.)$ satisfies triangle inequality from proposition 1. Thus $\forall i \in \{1, \ldots, f\}$ we have

$$\begin{aligned} d_S(\zeta_{i-1}, \zeta_f) &\leq d_S(\zeta_i, \zeta_f) + d_S(\zeta_{i-1}, \zeta_i) \\ d_S(\zeta_{i-1}, \zeta_f) - d_S(\zeta_i, \zeta_f) &\leq d_S(\zeta_{i-1}, \zeta_i) \end{aligned} \tag{4}$$

[5] All proofs are in the appendix.

[6] For a finite dataset the distance can however be zero between two different classifiers if they produce the same classification on it.

From Eq. 3 and given that $d_S(\zeta_{i-1}, \zeta_f)$ is an (unknown) constant at iteration i[7] we have,

$$\begin{aligned} Q_i^k &= \underset{q^k|q^k \in \mathcal{I}_{i-1}}{\text{argmin}}\ d_S(\zeta_i, \zeta_f) \\ &= \underset{q^k|q^k \in \mathcal{I}_{i-1}}{\text{argmax}}\ d_S(\zeta_{i-1}, \zeta_f) - d_S(\zeta_i, \zeta_f) \end{aligned} \tag{5}$$

However, since we do not know ζ_f, we maximize the upper bound of the function in Eq. 5 based on Eq. 4,

$$\bar{Q}_i^k = \underset{q^k|q^k \in \mathcal{I}_{i-1}^k}{\text{argmax}}\ d_S(\zeta_{i-1}, \zeta_i) \tag{6}$$

Now if we assume that the data is drawn independently and identically from some underlying distribution, then given the uncertainty in the missing values, we minimize the expectation of $d_S(\zeta_i, \zeta_f)$, which would lead us to query the following set based on the developments in this section

$$\begin{aligned} \hat{Q}_i^k &= \underset{q^k|q^k \in \mathcal{I}_{i-1}}{\text{argmax}}\ E[d_S(\zeta_{i-1}, \zeta_i)] \\ &= \underset{q^k|q^k \in \mathcal{I}_{i-1}}{\text{argmax}}\ \big(P(\zeta_{i-1}(x) \neq y, \zeta_i(x) = y) \\ &\quad + P(\zeta_{i-1}(x) = y, \zeta_i(x) \neq y)\big) \end{aligned} \tag{7}$$

since, the inequality in Eq. 4 is pointwise true and hence, $E[d_S(\zeta_{i-1}, \zeta_f) - d_S(\zeta_i, \zeta_f)] \leq E[d_S(\zeta_{i-1}, \zeta_i)]$. *Note that the computation of* $\hat{Q}_i^k$ *does not require knowledge of* ζ_f.

We can also derive an exponential bound on the probability of error reduction across successive runs, which is given in the following lemma.

Lemma 1 *Given a dataset S of size N, with* $d_S(.,.)$, ζ_i *and* ζ_f *defined as before, we have for* $t > p_i$,

$$P[d_S(\zeta_{i-1}, \zeta_f) - d_S(\zeta_i, \zeta_f) \geq Nt] \leq e^{-2N(t-p_i)^2} \tag{8}$$

where, $p_i = \frac{1}{N} E[d_S(\zeta_{i-1}, \zeta_i)]$.

While ideally we would want to maximize the left side probability, but since we do not know ζ_f we maximize the exponential bound. The bound is maximized when $p_i = \frac{1}{N} E[d_S(\zeta_{i-1}, \zeta_i)]$, is maximized and is thus consistent with what we surmised before. This idea of maximizing the distance between successive classifiers is the main motivation behind our algorithm in the next section. It is indeed useful that we were able to be move from a function that was dependent on ζ_f to a one that is independent of it.

Additionally, we seek to characterize the settings under which the solution obtained by maximizing the expected error reduction would be the same as maximizing the expected distance between consecutive classifiers. We now state a sufficient condition that ensures the same.

Proposition 2 *If we have a dataset S, and* ζ_i *at least correctly classifies those instances that were correctly classified by* ζ_{i-1} *where* $i \in \{1, \ldots, f\}$, *then the k instances that would be queried if we could maximize* $E[d_S(\zeta_{i-1}, \zeta_f) - d_S(\zeta_i, \zeta_f)]$ *in any iteration i are the same as those queried by maximizing* $E[d_S(\zeta_{i-1}, \zeta_i)]$.

[7] Since, ζ_{i-1} is known and ζ_f though not known is a fixed entity.

Algorithm 1 The proposed method MDIS to choose k instances to query

Input: $\mathcal{I}_0, C_0, k, \zeta_0, g, \mathcal{L}(.)$ {g is the maximum number of iterations and $\mathcal{L}$ is the learning algorithm.}
Output: $Q, \mathcal{C}$ {Sets of k instances queried at each iteration and the corresponding classifiers $\{\zeta_0, \ldots, \zeta_g\}$}

Let $Q = \phi, \mathcal{C} = \{\zeta_0\}$
for $i = 1$ to g **do**
  $\mathcal{S} = \phi, q_k = \phi$ {$\mathcal{S}$ stores the scores for each instance and q_k are the k instances to be queried.}
  $b = min(0.5 + \frac{k}{2|C_{i-1}|}, 1)$ {$|.|$ denotes cardinality. We are setting the bias for asymmetric uncertainty.}
  for all $(x, y) \in \mathcal{I}_{i-1}$ **do**
    $p = Prob(\mathcal{I}_{i-1}, C_{i-1}, \zeta_{i-1}(x), x, y)$ {Call $Prob()$ function in algorithm 2.}
    Compute $score_x = \frac{p(1-p)}{(-2b+1)p+b^2}$
    $\mathcal{S} = \mathcal{S} \cup score_x$
  end for
  Let $q_k = \{(x, y) : score_x$ is among the top/highest k in $\mathcal{S}\}$
  $Q = Q \cup q_k$
  $\mathcal{I}_i = \mathcal{I}_{i-1} - q_k$ and $C_i = C_{i-1} \cup q_k$
  $\zeta_i = \mathcal{L}(C_i)$
  $\mathcal{C} = \mathcal{C} \cup \zeta_i$
end for
Return $Q, \mathcal{C}$

Algorithm 2 $Prob(\mathcal{I}, C, \zeta, x, y)$: Routine to compute $P(\zeta_{i-1}(x) \neq y)$

Input: $F(.)$ {F is an univariate density estimator.}
Output: $P(\zeta(x) \neq y)$

Let $A = C \cup \{\hat{x} \in \mathcal{I} | \hat{x}^{M_{i-1}(x)} \subseteq T_{i-1}(\hat{x})\}$
{A is the set of all instances for whom the missing features in x are known.}

Compute $B = \{\zeta(\hat{x}) | \hat{x} \in A\}$
{Apply the classification function based on the missing features in x to all $\hat{x} \in A$ and obtain a set of values.}

Perform density estimation $F(B)$

Return $P(F(B) \leq v)$, where v depends on y and ζ and corresponds to $\zeta(x) \neq y$
{This will be clearer in Sect. 6, where we provide example instantiations to some popular classification techniques.}

The above proposition specifies a *sufficient condition* for the two solutions to match. This condition may however not be *necessary* for this to happen. Therefore, approximating or almost satisfying this condition where most but not all instances are correctly classified by ζ_{i-1} are now also correctly classified by ζ_i, may also lead to an optimal or at least close to the optimal for the left hand side. It is also useful to note that in practice, any reasonable classification algorithm will not take any extra effort to misclassify instances unnecessarily, and thus will not degrade the performance of our querying strategy even if the sufficient condition is not 100 % satisfied.

In the experimental section, we report how close we are to the satisfaction of this condition on several real datasets based on our algorithm presented in the next section. This will then provide us with an insight into the quality of our solution relative to the best possible case.

5 Method

As described in the previous section at any iteration i the idea is to query k instances such that ζ_i will be as different as possible from ζ_{i-1} with respect to our distance metric. Since we try to maximize the distance between successive classifiers we refer to our method as, maximum distance instance selection (MDIS). A prerequisite for the next classifier to be as different as possible from the current classifier is that the queried instances must encourage the classification algorithm to learn a different classifier or diverge from the current classifier.

When $k \ll |C_{i-1}|$, where $|.|$ denotes cardinality, any reasonable classification algorithm will in all likelihood learn a classifier ζ_i that is different from ζ_{i-1}, only if the queried k instances when completed are close to the boundary of ζ_{i-1} but are misclassified. Moreover, based on our strategy we also want the queried instances and if possible other misclassified instances to be correctly classified by the next classifier. Thus, we want to choose instances that have a misclassification probability above but close to 0.5. In the other two cases namely, if we choose instances that are likely to be correctly classified or if we choose instances that have a high probability of being misclassified, the classification algorithm is unlikely to change ζ_{i-1} thus not maximizing the distance. In the first case, since the k instances are likely to be correctly classified after completion by ζ_{i-1}, there is no incentive for the classification algorithm to learn a different ζ_i. In the second case, since the k instances are likely to be misclassified and far away from the boundary, which makes them hard to correctly classify, the classification algorithm will most likely give up on them as $k \ll |C_{i-1}|$ and therefore again not diverge from the current classifier. When k is comparable to $|C_{i-1}|$, we can be more ambitious and choose instances that are likely to be misclassified but farther away from the boundary. The reason being that the k instances are no longer insignificant in terms of the training error and hence the classification algorithm will learn a different classifier in an attempt to correctly classify them. However, here too we do not want to query extreme instances or outliers that are almost impossible to correctly classify, since based on our strategy we want the next classifier to be able to correctly classify them.

In Algorithm 1, we capture these intuitions by building a scoring function based on asymmetric uncertainty (Marcellin et al. 2006). This function ($score_x$ in Algorithm 1) is a smooth, strictly concave function defined over [0, 1]. It reaches a maximum at $b \in [0, 1]$ and is zero at the extremities of its domain i.e. when $P(\zeta_{i-1}(x) \neq y) = 0$ or when $P(\zeta_{i-1}(x) \neq y) = 1$. Therefore, when $b = 0.5$ it behaves like the standard uncertainty function and achieves its maximum at $P(\zeta_{i-1}(x) \neq y) = 0.5$. However, at other values of b the function shifts with the maximum occurring when $P(\zeta_{i-1}(x) \neq y) = b$. This is seen in Fig. 2, where $b = 0.7$ and hence, the function achieves its maximum at 0.7.

Given this and the intuitions described above to implement our strategy, we set $b = 0.5 + \frac{k}{2|C_{i-1}|}$. Typically, $k \leq |C_{i-1}|$ so in practice $b \leq 1$. This assignment of b ensures that we query instances that are likely to be misclassified and close to the boundary with us being more ambitious when k is comparable to $|C_{i-1}|$. Of course the multiplier 2 of $|C_{i-1}|$ is tunable depending on how ambitious one wants to be especially early on when $|C_{i-1}|$ is not too large. Another desirable property of this function is that it tapers down to zero faster on the side closer to the maximum, which should in all likelihood guard against choosing extremely hard to classify outliers making it again consistent with our strategy.

A simple illustration of this is seen in Fig. 3. With few complete examples and high b our method is more aggressive and chooses an instance to be completed that is highly likely to be currently misclassified. For low b, it chooses an instance with low misclassification probability, that is, one which is likely to be much closer to the decision boundary once

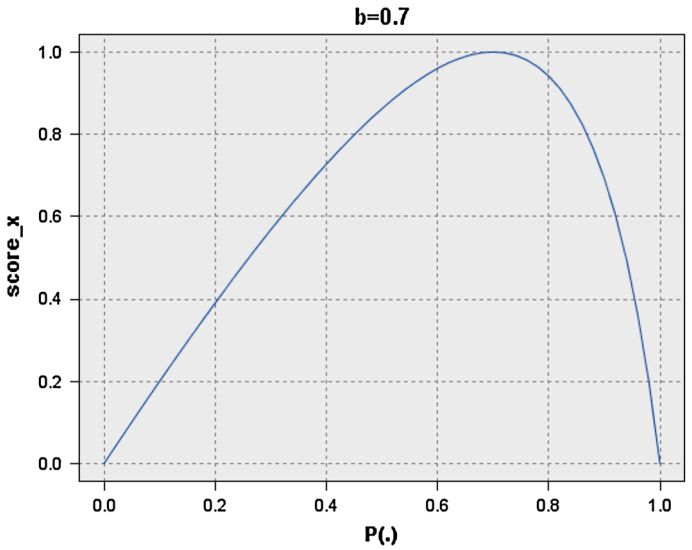

Fig. 2 Above we see our score function based on asymmetric uncertainty for $b = 0.s7$. We observe the behavior of the function for different values of $P(.) = P(\zeta_{i-1}(x) \neq y)$ (Color figure online)

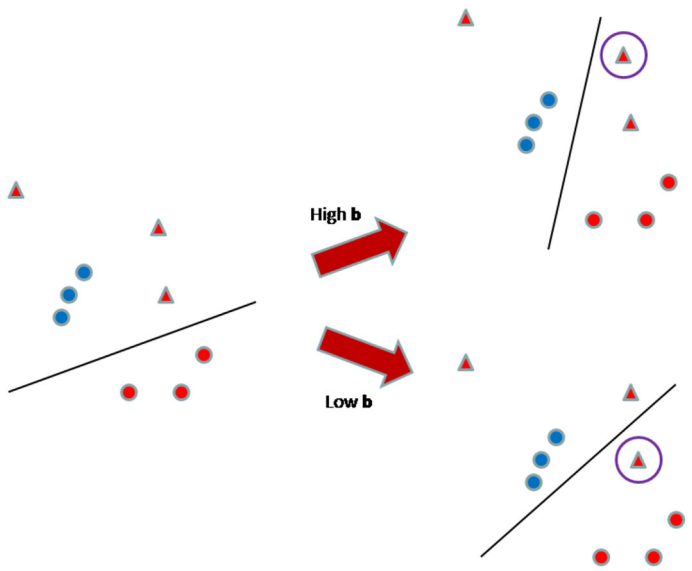

Fig. 3 Above we see an illustration of how our algorithm would behave for low and high b. The *black line* denotes the learned classifier. The *circles* are complete instances, while the triangles are incomplete ones. The *color* denotes the class label. The *purple circle* around the *triangle* denotes the instance that would be completed for $k = 1$ (Color figure online)

completed but on the incorrect side. In either case, our method is unlikely to choose the extreme outlier belonging to the red class, since our scoring function tapers rapidly to zero on the side of the maximum. As we query more and more instances the aggressiveness gets

moderated, since b monotonically reduces, with our method choosing instances that our likely to be close to the decision boundary and preferably misclassified.

Hence in general, our algorithm composes of the following steps. For a given k in any iteration i of our algorithm we:

- Compute the asymmetry parameter b.
- Compute $P(\zeta_{i-1}(x) \neq y)$ for each incomplete instance, i.e., the probability of incorrectly classifying it based on the current classifier as described in Algorithm 2.
- Compute the asymmetric uncertainty score for each incomplete instance based on b and $P(\zeta_{i-1}(x) \neq y)$.
- Query k instances with the highest score computed in the previous step.

6 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 the probability $P(\zeta_{i-1}(x) \neq y)$ for each incomplete $x \in \mathcal{I}_{i-1}$. Given the classification algorithm and the fact that the features are randomly (i.e. not consistently) missing for an instance we can compute this probability. The idea is to compute the functions $\zeta_{i-1}(x)$ by substituting the missing values in x by the corresponding values in $\hat{x}$. $\hat{x}$ is an instance for which we know values for at least those features that were missing in x. Thus, $\hat{x}$ could be complete or incomplete. We compute the function for each such $\hat{x}$ and then perform univariate density estimation on the corresponding values of the function. Based on this estimated density we can compute the above probability and consequently our score for x. The implicit assumption here is that for each incomplete x, we have a $\hat{x}$ for which the missing values in x are known. In other words, if we have more than 50 % features missing for each instance then our approach cannot be directly applied since, there will be no $\hat{x}$. We have seen however that this is not an unrealistic assumption in practice as in many cases we have large datasets with knowledge of most of the attributes, if not all, for at least a small fraction of instances. Moreover, we can always exclude instances for whom this condition is not met initially and query based on the rest. After just the first round of querying this issue, if it existed, would vanish since, we would have at least one complete instance.

Let us now see how the function looks for some popular classification techniques. Let $T_{i-1}(x)$ and $M_{i-1}(x)$ denote the set of features with known values and missing values for instance x prior to querying in iteration i respectively. Correspondingly, let $z^{T_{i-1}(x)}$ and $z^{M_{i-1}(x)}$ denote the set of entries i.e. features in instance z, whose values are known for x and missing for x in iteration i respectively.

6.1 Linear and non-linear SVMs

Support vector machines (Vapnik 1998) 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_{zl} denotes the dual variable in SVM optimization corresponding to instance z obtained by learning over C_l, which gives us the classifier ζ_l, then the classification function for x based on ζ_{i-1} is given by,

$$\zeta_{i-1}(x) : \sum_{\substack{(z,y_z)\\ \in C_{i-1}}} \alpha_{z(i-1)} y_z \mathcal{K}(z,x)$$
$$= \sum_{\substack{(z,y_z)\\ \in C_{i-1}}} \alpha_{z(i-1)} y_z \mathcal{K}\left(z^{T_{i-1}(x)} \cup z^{M_{i-1}(x)}, x^{T_{i-1}(x)} \cup x^{M_{i-1}(x)}\right) \tag{9}$$

where, $y_z \in \{1, -1\}$. Thus, we have a function in the missing features of x, i.e. in $x^{M_{i-1}(x)}$, since the other values are known. Now we can, as mentioned before, compute the function for instances where $M_{i-1}(x)$ are known and perform univariate density estimation using standard methods (Hastie et al. 2001). The above function can be instantiated for,

– Linear Kernels:

$$\mathcal{K}(z,x) = z \cdot x$$
$$= z^{T_{i-1}(x)} \cdot x^{T_{i-1}(x)} + z^{M_{i-1}(x)} \cdot x^{M_{i-1}(x)},$$

– Polynomial Kernels:

$$\mathcal{K}(z,x) = (z \cdot x)^a$$
$$= \left(z^{T_{i-1}(x)} \cdot x^{T_{i-1}(x)} + z^{M_{i-1}(x)} \cdot x^{M_{i-1}(x)}\right)^a,$$

– Radial Basis Kernels:

$$\mathcal{K}(z,x) = e^{-\gamma||z-x||^2}$$
$$= e^{-\gamma||z^{T_{i-1}(x)} - x^{T_{i-1}(x)}||^2} e^{-\gamma||z^{M_{i-1}(x)} - x^{M_{i-1}(x)}||^2}$$

or any other kernel that one wants to use.

Without loss of generality (w.l.o.g.) assume that the label of x is 1, i.e., $y = 1$, then our score for x based on the estimated density would be,

$$\frac{P(\zeta_{i-1}(x) \leq 0)(1 - P(\zeta_{i-1}(x) \leq 0))}{(-2b+1)P(\zeta_{i-1}(x) \leq 0) + b^2} \tag{10}$$

6.2 Logistic regression

Logistic regression (McCullagh and Nelder 1990) uses the logit link function to relate an input x to its output $Y = \{0, 1\}$ as follows,

$$ln\left(\frac{P(y=1)}{1 - P(y=1)}\right) = x\beta_l \tag{11}$$

where $y \in \{0, 1\}$ and β_l is the learned parameter vector over C_l. x is classified into class 1 iff $P(y = 1) \geq 0.5$. In this case, the classifier is given by,

$$\zeta_{i-1}(x) :$$
$$e^{x\beta_{i-1}} = e^{x^{T_{i-1}(x)} \beta_{i-1}^{T_{i-1}(x)}} e^{x^{M_{i-1}(x)} \beta_{i-1}^{M_{i-1}(x)}} \tag{12}$$

Given this and w.l.o.g. assuming $y = 1$ our score for x based on the estimated density would be,

$$\frac{P(\zeta_{i-1}(x) < 1)(1 - P(\zeta_{i-1}(x) < 1))}{(-2b+1)P(\zeta_{i-1}(x) < 1) + b^2} \tag{13}$$

These instantiations demonstrate how our strategy can be employed for standard classification techniques (Table 1).

Table 1 Information about the real datasets used in this study

Dataset	Source type	Source name	Size	Dimensionality	Attribute types
Credit approval	Public	UCI Rep.	690	15	Categorical/real
Spambase	Public	UCI Rep.	4601	57	Real
Adult	Public	UCI Rep.	48842	14	Categorical/real
Etoys	Private	Etoys	270	41	Categorical/real
Priceline	Private	Priceline	447	41	Categorical/real
Isolet	Public	UCI Rep.	7797	617	Real
Expedia	Private	Expedia	3125	41	Categorical/real

7 Experiments

We now compare our method MDIS with three state-of-the-art techniques namely; JIS, EU and ES. We do this on four UCI datasets *Credit Approval*, *Adult*, *Spambase* and *Isolet*, and three e-commerce datasets *Priceline*, *Etoys* and *Expedia*, some of which were used in prior works (Melville et al. 2005; Saar-Tsechansky et al. 2009; Sankaranarayanan and Dhurandhar 2013). The e-commerce datasets contain information about the online behavior of customers that visit the company's (retail) websites. The output variable in these datasets is indicative of a purchase being made during such a visit. The input features are a combination of browsing behavior and demographic information about the customers.

We perform 10 runs of 10 fold cross validation for the setting described next. For each instance in a dataset, we randomly assume 50 % of its input features to be missing. We incrementally complete instances in steps of 10 (i.e. $k = 10$) by querying them based on the decisions of the respective methods with the model being updated after each completion. This is the same experimental methodology as seen before (Sankaranarayanan and Dhurandhar 2013; Melville et al. 2005). We report the test accuracy at each completion averaged across all runs along with the 95 % confidence intervals as the measure of performance.

For our method we also show the extent to which our sufficient condition is satisfied. This gives us a sense of how close we are likely to be to the optimal solution. We do this by reporting the percentage of instances correctly classified by ζ_i that were correctly classified by ζ_{i-1} averaged (with a 95 % confidence interval) over the classifiers learned in the three ranges of completion; low (0, 33], medium (33, 66] and high (66, 100]. The ranges signify the percentages of completed instances relative to a dataset. Rather than averaging the results over the whole dataset this quantization provides a more granular view of how well our method is performing at different stages of completion.

We tested the different instance completion methods based on two baseline classification techniques namely, SVM with radial basis function (RBF) kernel and logistic regression. The density estimation for MDIS and JIS is done using the well known kernel density estimation (KDE) technique with Gaussian kernels. The SVM, logistic regression as well as the KDE used are part of Weka's implementation (Hall et al. 2009). We run the SVM with Weka's default parameters for γ and the slack variables. The KDE is performed assuming one Gaussian kernel per value used in the estimation (which is also the default) with precision set to 0.01.

We also performed experiments where the percentage of missing features was non-uniform, i.e., for one third of the instances we had 80 % randomly missing, for another

one third we had 50 % randomly missing and for the remaining one third we had 20 % randomly missing. As you will see later the results were qualitatively similar to the uniform case with our method performing significantly better than its competitors. We also experimented with varying k (5, 10, 20, 50) and varying the multiplier of $|C_{i-1}|$ that determines b, and found that our method is quite robust to these choices. We provide representative examples for these cases on the Credit Approval and Expedia datasets respectively, with SVM using RBF kernel as the baseline method.

7.1 Observations

From Fig. 4a–g, we see that our method MDIS consistently produces more accurate classifiers than the other methods across the different completion percentages. This remains unaltered even when we change the base classification method to logistic regression as is seen in Fig. 5a–g. Not only is the average accuracy high but the confidence intervals also do not overlap with the other methods for the most part making our method significantly better than the others in the statistical sense. It is especially encouraging to see that our method is significantly better than its competitors at low to moderate completion percentages, which is what we would usually encounter in many of these applications. At higher completion percentages, i.e. close to 100, almost all techniques will produce equally good classifiers since very few instances remain incomplete. This behavior is seen in the figures, where the gap in performance between the various methods generally reduces as most of the instances get completed.

In addition to evaluating the performance of our method relative to the other state-of-the-art methods, it is also interesting to get a feel for how close we are to the best possible solution. It is computationally too intensive to exhaustively search through all possible sets of $k = 10$ instances that give the best performance at each completion percentage on all the datasets. However, a good proxy for indicating the closeness of our solution to the optimal would be to study the extent to which our sufficient condition in proposition 2 is satisfied. As discussed before, approximately satisfying the sufficient condition might be good enough as this condition may not be *necessary* to obtain an optimal solution. Moreover, any reasonable classification algorithm will not take undue effort to misclassify instances that were previously correctly classified.

Given this, we observe in Figs. 4h and 5h the extent to which our sufficient condition is satisfied for the different datasets at low, medium and high completion percentages. 100 % satisfaction is of course ideal, but we see that in most cases the percentage is extremely close to 100. In fact, even the lowest is around 88 % for the Adult dataset at low completion percentages, which is promising. Therefore, we observe that even at low completion percentages our method is close to satisfying the sufficient condition and thus seems to be competitive with respect to the best case solution.

In Fig. 6, we see that even when the percentage of missing features is non-uniform our method remains significantly better than its competitors, with the results being qualitatively similar to the uniform case.

Figures 7 and 8, are an illustration of the fact that our method is robust to different values of k and b, although there are slight visible differences at low and intermediate completion percentages. Qualitatively similar results were seen on the other datasets.

From the above experimental results, we can see that not only is MDIS significantly better than the other state-of-the-art methods in the uniform and non-uniform (missing values) case but is also robust to different choices of k and the multiplier that determines b and is

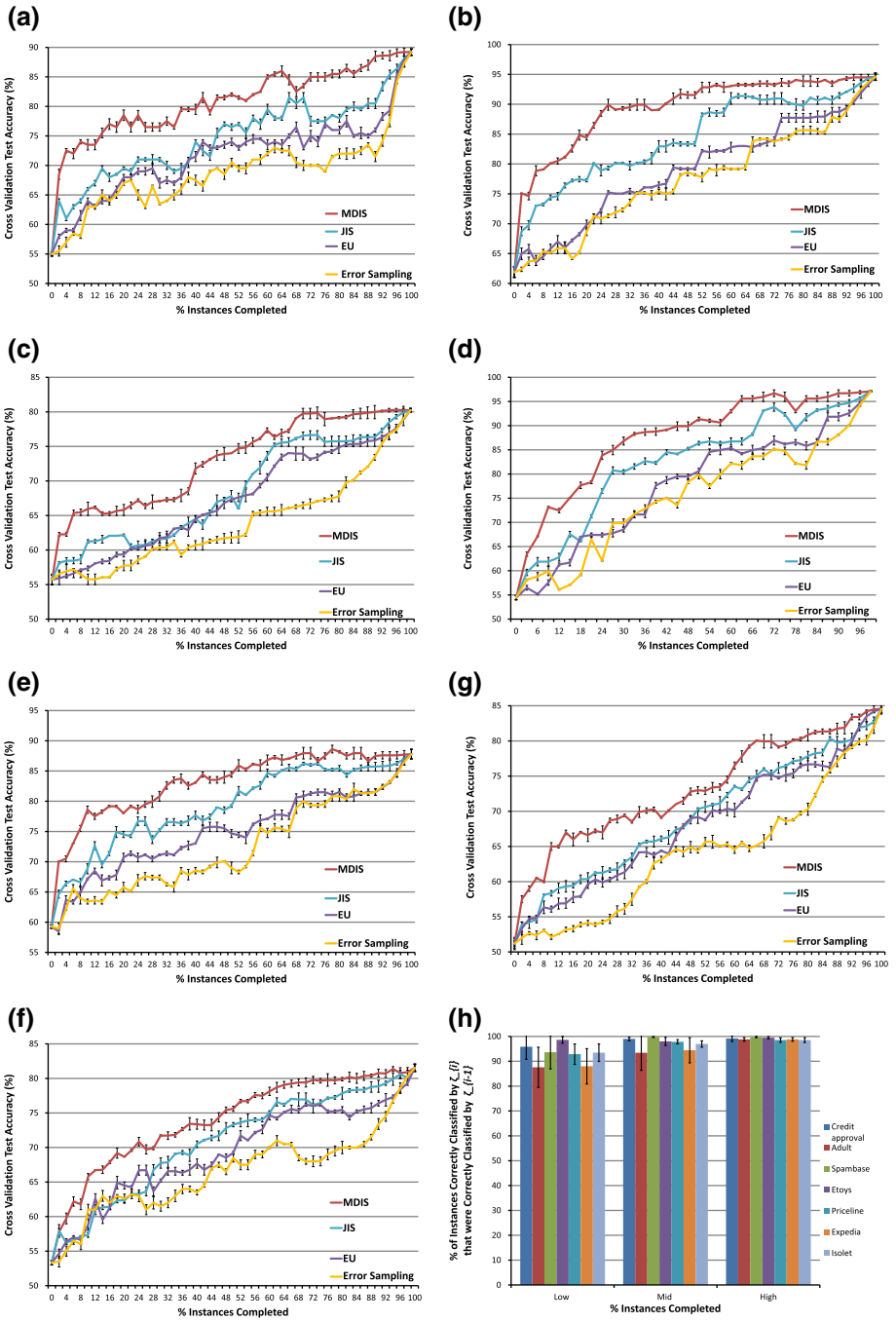

Fig. 4 The performance of the different methods on 7 real datasets, where the base classification method is SVM with RBF kernel. The last figure (**h**) depicts the extent to which our sufficient condition is satisfied. In particular, it shows the percentage of instances that ζ_i correctly classifies given that they were correctly classified by ζ_{i-1}, averaged over all such successive classifiers in the specified ranges. **a** Credit Approval, **b** Spambase, **c** Adult, **d** Etoys, **e** Priceline, **f** Expedia, **g** Isolet (Color figure online)

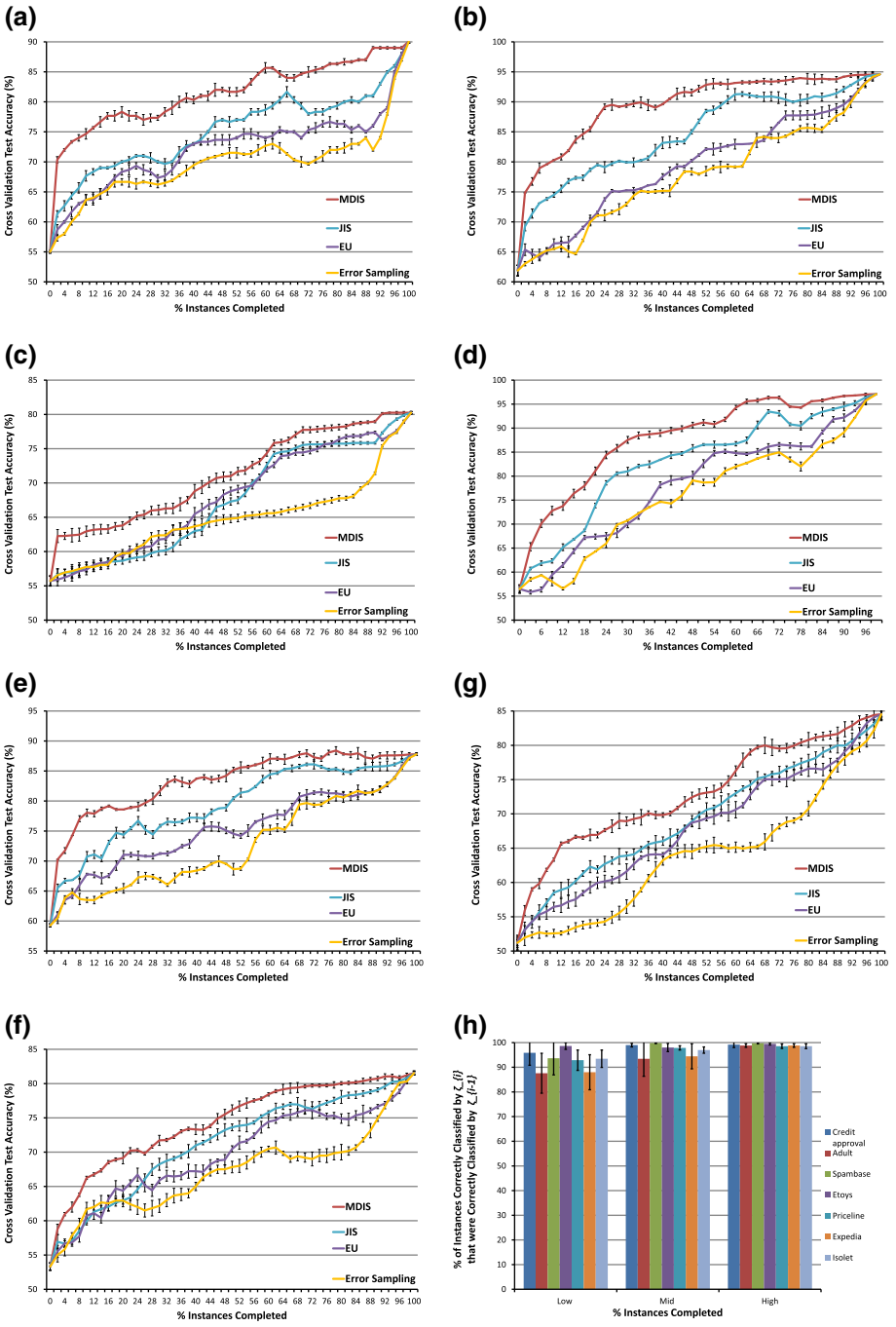

Fig. 5 The performance of the different methods on 7 real datasets, where the base classification method is logistic regression. The last figure (**h**) depicts the extent to which our sufficient condition is satisfied. In particular, it shows the percentage of instances that ζ_i correctly classifies given that they were correctly classified by ζ_{i-1}, averaged over all such successive classifiers in the specified ranges. **a** Credit Approval, **b** Spambase, **c** Adult, **d** Etoys, **e** Priceline, **f** Expedia, **g** Isolet (Color figure online)

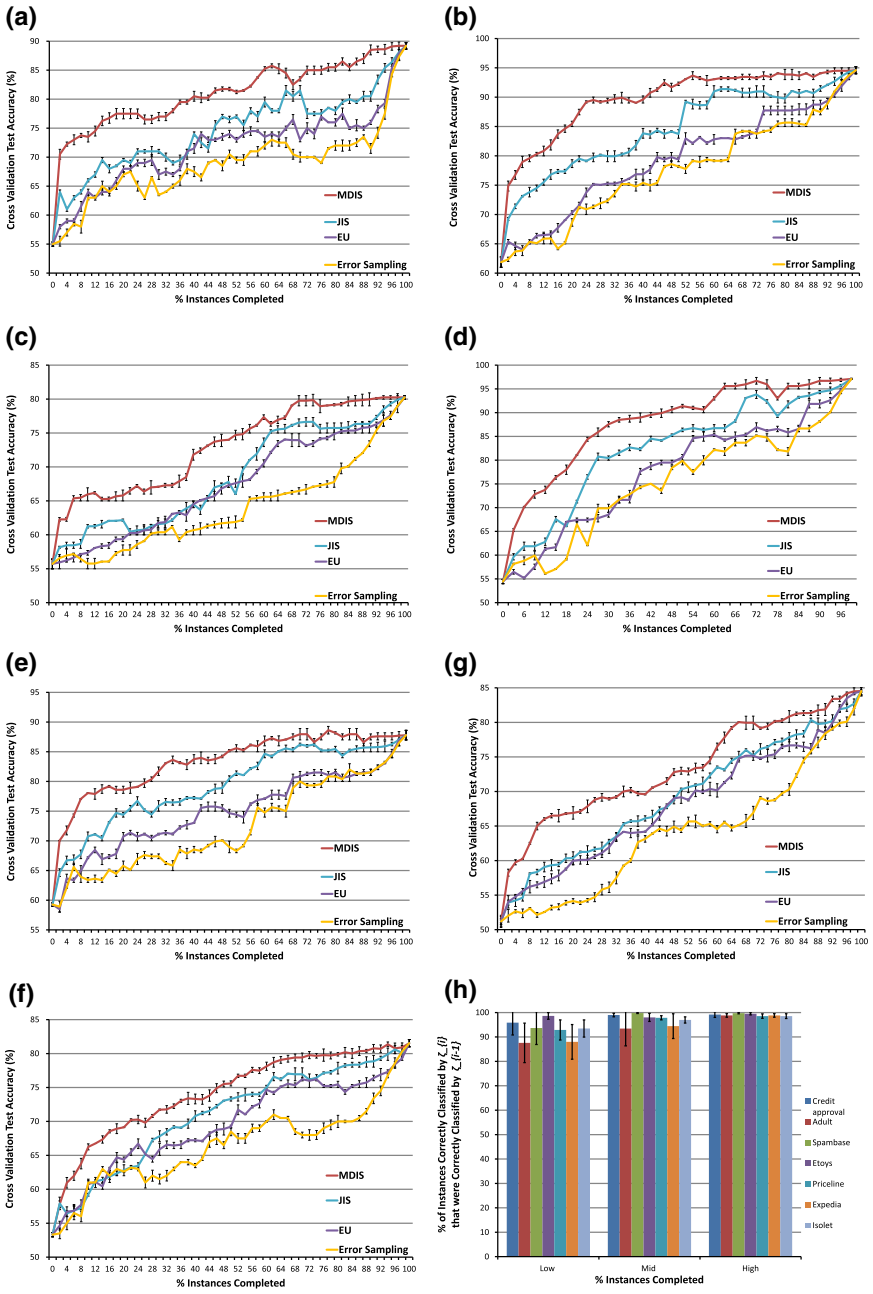

Fig. 6 The performance of the different methods on 7 real datasets, where the base classification method is SVM with RBF kernel and the percentage of missing values per instance is *non-uniform*. The last figure (**h**) depicts the extent to which our sufficient condition is satisfied. In particular, it shows the percentage of instances that ζ_i correctly classifies given that they were correctly classified by ζ_{i-1}, averaged over all such successive classifiers in the specified ranges. **a** Credit Approval, **b** Spambase, **c** Adult, **d** Etoys, **e** Priceline, **f** Expedia, **g** Isolet (Color figure online)

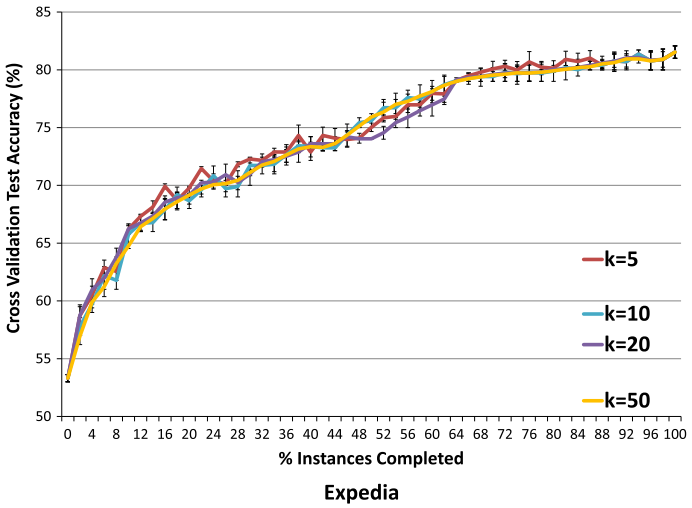

Fig. 7 The performance of our method on the Expedia dataset, where the base classification method is SVM with RBF kernel for different values of k (Color figure online)

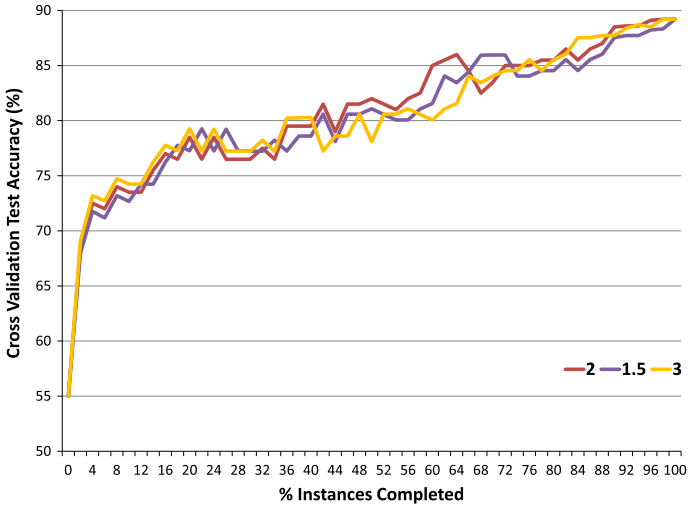

Fig. 8 The performance of our method on the credit approval dataset, where the base classification method is SVM with RBF kernel and the multiplier of $|C_{i-1}|$ which affects b is varied from 1.5 to 3 (Color figure online)

most likely close to the best possible solution relative to our distance function, which is a particularly encouraging observation.

8 Discussion

Some previous works have suggested incorporating a cost matrix κ, which indicates the cost of querying each entry. However, empirical studies in these works were performed on 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 keep the exposition clear. Nevertheless, the proposed method MDIS can in fact be easily extended to scenarios where this additional information is available.

If κ_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, $\kappa_i = \sum_{j \in M(x_i)} \kappa_i^j$. With this, the score for each instance x_i obtained from MDIS would just be scaled down by κ_i. Thus, if $score_{x_i}$ was the original score for x_i, the new score would be given by,

$$score_{x_i}^{new} = \frac{1}{\kappa_i} score_{x_i} \tag{14}$$

Our algorithm MDIS was one particular way of implementing our strategy of maximizing the distance between successive classifiers. Although it seems to be significantly better than the state-of-the-art methods in literature and possibly competitive with the optimal, it would be interesting to explore other approaches that implement our strategy in a more effective manner. Such approaches would probably pick support vectors earlier and outliers later than MDIS does. An approach worth exploring might be to find a proxy for the next classifier ζ_i and then query instances that this and the current classifier ζ_{i-1} disagree the most on. Finding a suitable proxy could be challenge, but one possibility could be to choose a classifier from the sequence of past classifiers $\{\zeta_0, \ldots, \zeta_{i-2}\}$ whose distance to the current classifier is the highest based on the current completed set. Another possibility could be to assume a distribution over the classifiers and sample the proxy for the next classifier from this distribution conditioned on the previous classifiers.

Alternatively, one may define other distance metrics possibly by restricting the hypothesis space viz. angles for linear hypothesis, and provide algorithms that work well in these settings.

In summary, we proposed a novel framework for the instance completion problem. Based on this framework we proposed a new meta-technique, which as we have shown can be easily instantiated for popular classification algorithms. Computationally, it still has the benefit of JIS, which requires only univariate probability estimation, although they both consider a function over all the missing features of an instance, thus analyzing the features jointly rather than independently as is the case for some other methods. We have also witnessed the strength of our technique, in the experimental section, where it is much superior in performance compared with other state-of-the-art methods. Moreover, based on the extent of satisfaction of our sufficient condition we have seen that our method in all likelihood is also competitive with the optimal method which would minimize our distance to the final classifier.

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

Appendix

Proof of Proposition 1

Proof It is easy to see that $d_S(.,.) \geq 0$ as it is the sum of non-negative functions. It also easy to see that it is symmetric as our loss functions are symmetric in the first two arguments which are the arguments of $d_S(.,.)$ and the sum of symmetric functions is a symmetric function.

The argument for triangle inequality is more interesting. Consider 3 classifiers ζ_1, ζ_2 and ζ_3. W.l.o.g. assume that $\forall i \in \{1, 2\}, d_S(\zeta_1, \zeta_2) \geq d_S(\zeta_i, \zeta_3)$, i.e., ζ_1 and ζ_2 are the farthest away relative to dataset S. Let $T \subseteq S$ such that $\forall (x, y) \in T, \lambda(\zeta_1(x), \zeta_2(x), y) = 1$ and $\forall (x, y) \in S/T, \lambda(\zeta_1(x), \zeta_2(x), y) = 0$ i.e., the set of instances in S that ζ_1 or ζ_2 classifies correctly. With this we have,

$$\begin{aligned} & d_S(\zeta_1, \zeta_2) \\ &= \sum_{(x,y)\in T} \lambda(\zeta_1(x), \zeta_2(x), y) \\ &= \sum_{(x,y)\in T} (\lambda(\zeta_1(x), \zeta_3(x), y) + \lambda(\zeta_2(x), \zeta_3(x), y)) \\ &\leq \sum_{(x,y)\in S} (\lambda(\zeta_1(x), \zeta_3(x), y) + \lambda(\zeta_2(x), \zeta_3(x), y)) \\ &= d_S(\zeta_1, \zeta_3) + d_S(\zeta_2, \zeta_3) \end{aligned}$$

The second equality in the above equation comes from the fact that ζ_3 will either correctly or incorrectly classify an instance and since either ζ_1 or ζ_2 (but not both) correctly classify instances in T, ζ_3 will have a loss of 1 with exactly one of them. The next steps follow easily and, since as per our assumption ζ_1 and ζ_2 are the farthest, we have our proof. □

Proof of Lemma 1

Proof Given a dataset S of size N, with $d_S(.,.)$, ζ_i and ζ_f defined as before, we have for $t, h > 0$

$$\begin{aligned} & P[d_S(\zeta_{i-1}, \zeta_f) - d_S(\zeta_i, \zeta_f) \geq Nt] \\ &\leq P[d_S(\zeta_{i-1}, \zeta_i) \geq Nt] \\ &\leq E[e^{h(d_S(\zeta_{i-1},\zeta_i)-Nt)}] \\ &= e^{-hNt} E[e^{hd_S(\zeta_{i-1},\zeta_i)}] \\ &= e^{-hNt} \prod_{j=1}^{N} E[e^{h\lambda(\zeta_{i-1}(x_j),\zeta_i(x_j),y_j)}] \\ &= U \end{aligned}$$

Let λ_j denote $\lambda(\zeta_{i-1}(x_j), \zeta_i(x_j), y_j)$. Now from Jensens inequality we have, $e^{h\lambda_j} \leq \lambda_j(e^h - 1) + 1$. Thus,

$$E[e^{h\lambda_j}] \leq p_i(e^h - 1) + 1$$

where, $p_i = E[\lambda_j] = \frac{1}{N} E[d_S(\zeta_{i-1}, \zeta_i)]$.

Now let $l(h) = ln(p_i(e^h - 1) + 1)$. By second order Taylor expansion around zero, we get $l(h) \leq \frac{1}{8}h^2 + p_i h$. Thus,

$$U \leq e^{-hNt + \frac{1}{8}Nh^2 + Nhp_i} \tag{15}$$

Minimizing the above equation w.r.t. h and setting it to zero, we get the optimal value of $h = 4(t - p_i)$.

Now substituting this value of h in Eq. 15, we get,

$$U \leq e^{-2N(t-p_i)^2}$$

which is our result. □

Proof of Proposition 2

Proof Considering the left side of the inequality i.e. the error reduction part, we would query instances that will lead to the greatest reduction in error, since ζ_f is the classifier with lowest error. Given our assumption of ζ_i correctly classifying at least those instances that were correctly classified by ζ_{i-1}, mathematically, we would query instances such that $P(\zeta_{i-1}(x) \neq y, \zeta_i(x) = y)$ is maximized.

Considering the right side of the inequality, we want to query instances that would maximize the distance between ζ_{i-1} and ζ_i. Unconstrained this would be the set that would lead to ζ_i flipping the predictions of ζ_{i-1} to the correct/incorrect class. Mathematically, this would be equivalent to querying instances such that $P(\zeta_{i-1}(x) \neq y, \zeta_i(x) = y) + P(\zeta_{i-1}(x) = y, \zeta_i(x) \neq y)$ is maximized. However, we assume that ζ_i classifies correctly at least those instances that were correctly classified by ζ_{i-1}, which implies

$$\begin{aligned} &P(\zeta_{i-1}(x) = y, \zeta_i(x) \neq y) \\ &= P(\zeta_i(x) \neq y | \zeta_{i-1}(x) = y) P(\zeta_{i-1}(x) = y) \\ &= 0 \end{aligned}$$

since $P(\zeta_i(x) \neq y | \zeta_{i-1}(x) = y) = 0$. Hence, in this case too we would query instances that would maximize $P(\zeta_{i-1}(x) \neq y, \zeta_i(x) = y)$, which is identical to the left side. □

References

Bilgic, M., & Getoor, L. (2007). Voila: Efficient feature-value acquisition for classification. In *AAAI '07: Proceedings of the 22nd national conference on artificial intelligence*.

Bilgic, M., & Getoor, L. (2011). Value of information lattice: exploiting probabilistic independence for effective feature subset acquisition. *Journal of Artificial Intelligence Research*, *41*(2), 69–95.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Dasgupta, S., & Langford, J. (2009). Active learning tutorial. In *Proceedings of the 26th international conference on machine learning*.

Desjardins, M., Macglashan, J., & Wagstaff, K. (2010). Confidence-based feature acquisition to minimize training and test costs. In *Proceedings of SIAM conference on data mining*.

Frazier, P. (2009). *Knowledge-gradient methods for statistical learning*. Princeton: Princeton University Press.

Guo, Y., & Schuurmans, D. (2007). Discriminative batch mode active learning. In *NIPS*. Curran Associates Inc.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Exploration Newsletter*, *11*(1), 10–18.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques*. Amsterdam: Elsevier.

Hanneke, S. (2007). A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th international conference on machine learning*.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. Berlin: Springer.

Hoi, S. C. H., Jin, R., Zhu, J., & Lyu, M. R. (2006). Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on machine learning*.

Kanani, P., & Melville, P. (2008). Prediction-time active feature-value acquisition for customer targeting. In *Proceedings of the workshop on cost sensitive learning, NIPS 2008*.

Kapoor, A., & Greiner, R. (2005). Learning and classifying under hard budgets. In *Proceedings of the European conference on machine learning* (pp. 170–181). Springer.

Lewis, D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In: *Proceedings of the eleventh international conference on machine learning* (pp. 148–156). Morgan Kaufmann.

Lizotte, D. J., & Madani, O. (2003). Budgeted learning of naive-bayes classifiers. In *Proceedings of 19th conference on uncertainty in artificial intelligence (UAI-2003* (pp. 378–385). Morgan Kaufmann.

Marcellin, S., Zighed, D., & Ritschard, G. (2006). Detection of breast cancer using an asymmetric entropy measure. In *Proceedings of computational statistics*.

McCullagh, P., & Nelder, J. (1990). *Generalized linear models* (2nd ed.). Boca Raton: Chapman and Hall.
Melville, P., Saar-Tsechansky, M., Provost, F., & Mooney, R. (2004). Active feature-value acquisition for classifier induction. In *Proceedings of the fourth IEEE international conference on data mining*. IEEE Computer Society.
Melville, P., Saar-Tsechansky, M., Provost, F., & Mooney, R. (2005). An expected utility approach to active feature-value acquisition. In *Proceedings of the fourth IEEE international conference on data mining*. IEEE Computer Society.
Moon, S., McCarter, C., & Kuo, Y.-H. (2014). Active learning with partially featured data. In *Proceedings of the 23rd world wide web companion conference*.
Saar-Tsechansky, M., Melville, P., & Provost, F. (2009). Active feature-value acquisition. *Management Science*, *55*(4), 664–684.
Saar-Tsechansky, M., & Provost, F. (2004). Active sampling for class probability estimation and ranking. In *Machine learning* (pp. 153–178).
Sankaranarayanan, K., & Dhurandhar, A. (2013). Intelligently querying incomplete instances for improving classification performance. In *Proceedings of the 22Nd ACM international conference on information and knowledge management*.
Settles, B. (2010). *Active learning literature survey*. Technical report.
Sheng, V. S., & Ling, C. X. (2006). Feature value acquisition in testing: A sequential batch test algorithm. In *Proceedings of international conference on machine learning* (pp. 809–816).
Vapnik, V. (1998). *Statistical learning theory*. London: Wiley.
Wang, Z., & Ye, J. (2013). Querying discriminative and representative samples for batch mode active learning. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining*.
Weiss, S. M., Dhurandhar, A., & Baseman, R. J. (2013). Improving quality control by early prediction of manufacturing outcomes. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining*.
Zheng, Z., & Padmanabhan, B. (2002). On active learning for data acquisition. In *Proceedings of the fourth IEEE international conference on data mining*. IEEE Computer Society.