Artificial Intelligence in Medicine xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

journal homepage: www.elsevier.com/locate/aiim

A decision support system for cost-effective diagnosis

Chih-Lin Chi [a,*], W. Nick Street [b], David A. Katz [c]

[a] Center for Biomedical Informatics, Harvard Medical School, 10 Shattuck Street, Boston, MA 02115, USA
[b] Management Sciences Department and Interdisciplinary Graduate Program in Informatics, S232 Pappajohn Business Building, The University of Iowa, Iowa City, IA 52242, USA
[c] University of Iowa Carver College of Medicine and Center for Research in the Implementation of Innovative Strategies in Practice, VA Medical Center, 601 Hwy 6 West, Mailstop 152, Iowa City, IA 52246, USA

ARTICLE INFO

Article history:
Received 1 March 2009
Received in revised form 23 April 2010
Accepted 16 August 2010

Keywords:
Decision support systems
Machine learning
Optimization
Feature selection
Utility-based data mining
Cost-effective diagnosis

ABSTRACT

Objective: Speed, cost, and accuracy are three important goals in disease diagnosis. This paper proposes a machine learning-based expert system algorithm to optimize these goals and assist diagnostic decisions in a sequential decision-making setting.
Methods: The algorithm consists of three components that work together to identify the sequence of diagnostic tests that attains the treatment or no test threshold probability for a query case with adequate certainty: lazy-learning classifiers, confident diagnosis, and locally sequential feature selection (LSFS). Speed-based and cost-based objective functions can be used as criteria to select tests.
Results: Results of four different datasets are consistent. All LSFS functions significantly reduce tests and costs. Average cost savings for heart disease, thyroid disease, diabetes, and hepatitis datasets are 50%, 57%, 22%, and 34%, respectively. Average test savings are 55%, 73%, 24%, and 39%, respectively. Accuracies are similar to or better than the baseline (the classifier that uses all available tests in the dataset).
Conclusion: We have demonstrated a new approach that dynamically estimates and determines the optimal sequence of tests that provides the most information (or disease probability) based on a patient's available information.

1. Introduction

Diagnostic decision making is based on experience and hypothetico-deductive reasoning [1]. In the face of diagnostic uncertainty, a physician can either gather more evidence (test) or treat a patient [2,3]. According to Bayesian theory, the clinician adjusts the likelihood of the disease in question with each new diagnostic test. The desired level of certainty depends largely on the consequences of inaccurate diagnosis; one generally needs to perform diagnostic tests until one has attained a treatment (or no test) threshold probability, i.e., the threshold of sufficient evidence [1,2].

Each test result can revise the probability of disease in relation to the treatment (or no-test) threshold, but the informativeness of the same test result may vary based on the pre-test (prior) probability, which in turn varies as a function of patient characteristics and other test results. In addition, test parameters, such as sensitivity and specificity may vary across different patient populations [4–7].

For a given individual or patient subgroup, it would be highly desirable to identify the test sequence among all candidate sequences that optimizes the confidence of decision making while minimizing cost. In advance of knowing the possible test results for a given individual, one would like to identify the test whose result is most likely to approach the treatment (or no-test) threshold.

This paper proposes a machine learning (ML)-based expert system approach, called optimal decision path finder (ODPF), to dynamically determine the test that is most likely to be informative in terms of diagnostic accuracy while minimizing the time and money spent on diagnostic testing. Two types of tests are considered: immediate and delayed tests. The first type of test such as blood pressure is routine and inexpensive, and the results can be known immediately. The second type of test is more costly, and the test results are not immediately available. This research focuses on the second type of test. Our algorithm takes pre-test probability, interaction of variables, and the cost of each test into account and uses a greedy search to guess the test and generate an individualized test sequence.

2. Background

2.1. Expert systems

As ODPF is an automated decision-making process, we place it in the familiar context of expert systems. An expert system usually

* Corresponding author. Tel.: +1 617 432 7185; fax: +1 617 432 6675.
E-mail addresses: Chih-Lin_Chi@hms.harvard.edu, chih-lin-chi@uiowa.edu (C.-L. Chi).

0933-3657/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.artmed.2010.08.001

consists of a knowledge source and a mechanism for problem solving that returns a response based on the information provided by the query. The knowledge source of most expert systems (e.g., knowledge-based systems (KBS), fuzzy expert systems) is based on direct input from domain experts and evidence from the literature. As an early example, MYCIN [8] provides diagnostic and therapeutic recommendations. The knowledge in MYCIN is stored in the form of rules, which were elicited from infectious disease experts. The process of transforming human knowledge to machine-usable form is called knowledge acquisition and is considered a bottleneck because it is time- and labor-intensive [9]. In addition, maintaining the knowledge base is very labor-intensive [10,11].

Other systems use techniques such as case-based reasoning and machine-learning methods for inference, and are thus based exclusively on data. They can avoid the knowledge acquisition problem, e.g., case-based reasoning (CBR) as described in [12]. In CBR, the knowledge consists of previous cases, including the problem, the solution, and the outcome, stored in a central location, called the case library. To obtain the solution for a new case, one simply identifies the case that is most similar to the problem in the case library, and the proposed solution can be adapted from the retrieved case.

ML methods use mathematical equations, rules, or decision trees to represent the decision function. Using a mathematical form of knowledge has the advantages of stability, observability, and controllability [13,14]; however, complex real-world phenomena cannot always be modeled using mathematical equations. When the phenomenon of interest can be accurately modeled, we can use several optimization approaches to guide actions. For example, Chi et al. devised an expert system to select the hospital in which a given patient can obtain the highest probability of survival and lowest probability of complications [15]. In another example, Liau et al. [16] devised an expert system in a crude oil distillation unit to help control parameters to maximize oil production rate.

Similar to case-based systems, ML-based expert systems can avoid the bottleneck of knowledge acquisition because knowledge is directly obtained from data. In addition, ML-based expert systems are able to give recommendations that are generated by non-linear forms of knowledge, and are easily updated by simply adding new cases. This paper shows an application of an ML-based expert system that uses a non-linear form of knowledge and optimization techniques to guide selection of diagnostic testing sequences.

2.2. Machine learning

Inductive machine learning algorithms can learn patterns from labeled data, i.e., cases that have a known outcome [17]. The decision functions, which result from the training process, can predict labels (or scores) based on a set of input variables or features and represent the knowledge that is mined from the data. With an appropriate design, one can apply these functions to many applications, such as word recognition, movie recommendations, etc. Here we briefly introduce background for specific ML research areas incorporated into our system.

2.2.1. Feature selection/acquisition

The purpose of feature selection is to reduce the number of predictive features, reducing data-collection costs while either improving or not affecting predictive performance in unseen cases. Feature selection techniques can be divided into filter and wrapper models [18]. The filter model is a preprocessing step that occurs prior to the induction method. Feature ranking based on correlation with the class label (i.e., outcome) is an example of filtering. In a wrapper model, the induction method is used to evaluate candidate feature subsets (e.g., forward selection, backward elimination [19]). Another group of feature selection techniques use embedded methods [19,20], in which the optimization of model construction directly includes a penalty term corresponding to the number of predictive features.

Most feature selection methods select one set of features for all data points, which is a global strategy. Local variants find multiple sets of features for different data points. This local strategy examines local regions of the feature space, as the relevance of features may vary across different clusters of cases [21]. The most extreme form of local strategy uses one set of features for each instance. For example, Domingos [22] uses a clustering-like approach to select sets of locally-relevant features. The feature-weighted methods also use different weighted sets for different instances or classes. For example, Howe and Cardie [23] use class distribution weighting to compute a different weight vector for each class, and Park et al. [24] use artificial neural networks (ANNs) to compute the feature weights for each instance.

Our approach is also related to utility-based data mining, including such methods as active and cost-sensitive learning. For example, some active learning studies [25–28] automatically acquire feature values, instances, and/or class labels to reduce the cost of data collection and obtain maximum information. Cost-sensitive learning [29] was originally formulated to treat misclassification of different classes unequally and uses different costs to construct the model. Studies such as those by Turney [30] and Ling et al. [31] aggregate both misclassification and attribute costs, explicitly addressing the trade-off between cost and performance.

Previous studies in the medical decision making literature [32–34] also integrate the misclassification costs of different classes with test costs, and make a diagnosis when the cost of testing exceeds the value of additional information. Our use of thresholds has a similar function to various misclassification costs, because one can set various thresholds for different classes to control misclassification of disease. In order to achieve the goal of quick diagnosis, ODPF uses feature selection to identify the test that is most likely to confirm diagnosis (or cross a threshold).

Existing local feature selection techniques can select relevant features for an instance, in which the order of including features is not a concern. For the sequential diagnosis problem, we assume that a query is a new patient, such that each medical test can be seen as a feature whose value is initially unknown. Feature-selection techniques may identify at the outset all important features that one needs to characterize a patient. In the sequential diagnosis scenario, one needs to select tests sequentially and dynamically because one needs to determine whether a diagnosis can be made with confidence or whether more evidence is needed, especially when the test is very expensive. In addition, various feature subsets may result in different optimal selection of a feature, i.e., the optimum selection of a feature may be influenced by features at hand [35]. Thus, recursively including one test at a time may be better than including all relevant tests simultaneously.

In this paper, we propose a new feature selection technique called locally sequential feature selection (LSFS). LSFS is a local feature selection technique for one instance (query patient) that recursively determines the most relevant feature based on the current subset of features and their values. In other words, the next test is determined based on the available information, e.g., symptoms and previous test results. Once the treatment threshold is reached, the LSFS algorithm stops. The criteria used to select a test include the relevance of a feature and the associated cost; these criteria are implemented in speed-based and cost-based objective functions, which will be discussed in Section 3.3.

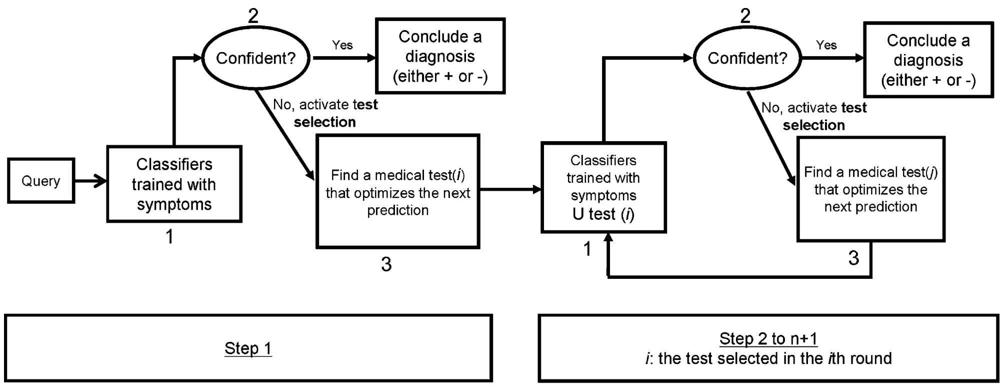
2
Confident?
Yes
Conclude a diagnosis (either + or -)
No, activate test selection
Query
Classifiers trained with symptoms
1
Find a medical test(i) that optimizes the next prediction
3
Classifiers trained with symptoms U test (i)
1
2
Confident?
Yes
Conclude a diagnosis (either + or -)
No, activate test selection
Find a medical test(j) that optimizes the next prediction
3
Step 1
Step 2 to n+1
i: the test selected in the ith round

Fig. 1. Summary of ODPF application.

2.2.2. Lazy learning

Lazy learning – also called instance-based, case-based, memory-based learning – builds a prediction model specifically for the query case. For example, a *k*-nearest neighbor (*k*-NN) classifier finds the *k* closest training cases to identify the best label for the query case. Lazy learning algorithms show three types of properties [36]. First, the classifiers defer processing of the output until a query case appears. Second, their responses combine the training with the query information. Third, they discard the constructed answer and any intermediate results. In contrast to lazy learning algorithms, eager learning algorithms, such as ANNs and decision trees, compile the data in advance and use it to construct a predictive model. They use this global model to give responses to all queries.

An alternative form of lazy learning is locally-weighted learning [37]. Instances are weighted based on the distance to a query point. This study turns support vector machines (SVMs) into lazy SVMs as the base learners (e.g., *d* in Section 3.4), by assigning weights to instances. The advantage of lazy SVM is that the base learners can be adapted for the query, i.e., the more test results we know, the more the classifiers can be trained to focus on data points similar to the current query.

3. Methods

Fig. 1 shows a visual representation of the diagnosis process that is supported by ODPF algorithm. The process begins with the basic information of a query patient, and ends with a confident diagnosis—disease or no disease. The question of what constitutes a “confident” diagnosis is determined by two (initially, symmetric) thresholds, a treatment threshold (above which disease is highly likely) or a no-test threshold (below which disease is highly unlikely and testing can be stopped) (Fig. 2). The thresholds must be determined by physicians or other experts, based on an analysis of relative costs and benefits [2].

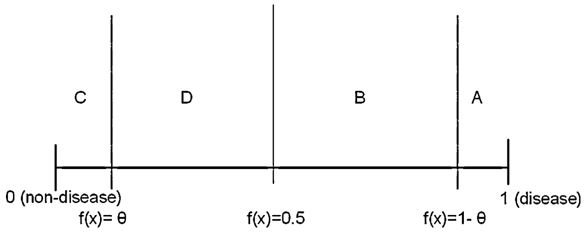

Fig. 2. Confident diagnosis (prediction). $f(x) = 0.5$) represents a separation surface, which can distinguish presence from absence of disease. A prediction with $f(x) > 0.5$ indicates a positive diagnosis (disease is present). There are two confidence thresholds $[1 - \theta, \theta]$ for a decision of treatment or no-test. A diagnosis can be made only when the prediction of a patient’s disease status crosses the threshold (area A or C). Otherwise, one needs more testing to support the prediction (area B or D).

Examining the confidence level of the diagnosis and selecting the next test are repeated in each step. Initially (step 1), we only have the patient’s symptoms and/or some immediately available test results. A lazy classifier is trained with these data (performed by module 1) and predicts whether or not a given patient has the disease of interest. The system then examines the prediction in relation to the treatment and no-test thresholds (performed by module 2). If the prediction is sufficiently confident, the system makes a diagnosis, and the process terminates. Otherwise, the system looks for a test that can facilitate the prediction in the next step (performed by module 3), and we do not know the test result until step 2.

In module 1 at step 2, after the result of the selected test (determined by module 3 in step 1) has been obtained for the query, we train a new classifier with symptom data and the values of the selected test (*test*(*i*)) from the training cases. (We note that we need to train a query-specific classifier, so we build a classifier with features corresponding to the query, i.e., symptoms and (*test*(*i*)) Then we examine the confidence of the prediction (performed by module 2) on the query and, if necessary, select another new test (performed by module 3). These two steps are repeated until a confident diagnosis occurs or until all options for testing have been exhausted, at which point a diagnosis is made. If no confident diagnosis can be made using all the tests, the probability of disease estimated by the final classifier is presented to the physician.

We detail lazy learning, LSFS, inheritance strategies, missing values, multi-class prediction, unbalanced data, and description of datasets in the following subsections.

3.1. Lazy support vector machines (SVMs)

SVMs [38] are a popular predictive model that can avoid overfitting problems. Eq. (1) shows the primal optimization model for training an SVM classifier. A training case is represented by a vector of its predictive features x_i and its diagnostic outcome y_i, which is either 1 or -1. W is the weight vector, and b is the bias term for the decision function. ε_i is the error of training case i, and C is a given constant that controls the balance between error and model sparsity.

$$\begin{aligned} \underset{W,\varepsilon}{\text{minimize}} \quad & \langle W \cdot W \rangle + C\left(\sum_{i=1}^{l} \varepsilon_i\right) \\ \text{subject to} \quad & y_i(\langle W \cdot x_i \rangle + b) \geq 1 - \varepsilon_i, i = 1, \ldots, l \\ & \varepsilon_i \geq 0, i = 1, \ldots, l \end{aligned} \qquad (1)$$

A typical implementation of SVMs solves the dual of this problem and induces nonlinearity in the separating surface using a kernel function (see Vapnik [38] for details).

$$\begin{aligned} &\underset{W,\varepsilon}{\text{minimize}} && \langle W \cdot W \rangle + \left(\sum_{i=1}^{l} C_i \varepsilon_i \right) \\ &\text{subject to} && y_i(\langle W \cdot x_i \rangle + b) \geq 1 - \varepsilon_i, i = 1, \ldots, l \\ & && \varepsilon_i \geq 0, i = 1, \ldots, l \end{aligned} \tag{2}$$

In this project, we use lazy SVMs, which learn a decision function specifically for the query case, as the base learners. Eq. (2) shows the modified optimization formulation when training a lazy SVM. The only difference between Eqs. (2) and (1) is C. In Eq. (2), the value of C is different for each training case i. When C_i is large for the training case i, case i gets more attention when optimizing, reducing the chance of error for case i. In both equations, $(\langle W \cdot x_i \rangle + b)$ is the decision function $d(x)$ that classifies whether or not a patient has the disease of interest. In $d(x)$, W represents the degree of importance of variables and b represents the bias term.

The value of C_i is determined by the similarity to the query, which is solely based on the results of the medical tests. An extreme example is when all test values of a training case are exactly the same as the query; this training case will have a very large C_i. On the other hand, when a case has no test value matching the query, the case will have a small C_i.

We use a simple algorithm to update instance weight (C_i) as illustrated in Fig. 3, which shows a query and three training cases. γ ($\geq$1) is the multiplier for updating instance weights. The initial weight for all training cases is 1 because we do not have any prior knowledge of these training cases. In step 1, the query does not pass the confidence threshold and we select Test X. After the test result has been determined, only training case 3 has the same value as the query. Thus, its instance weight (C_3) is multiplied by a factor of γ (in this example, we use $\gamma = 2$). In step 2, the values of training cases 2 and 3 have the same value as the query, and their instance weights (C_2 and C_3) are multiplied by 2. In step 3, only C_2 and C_3 are multiplied again by 2.

After the selected test result has been obtained in each step, we can update the instance weights for all training cases. More similar cases have higher weight, making the training case population more specific to the query after more tests results are known. In our experiments, the multiplier γ is decided empirically based on predictive performance and costs.

3.2. Predicting probabilities

Although SVM can classify well, the output scores are not probabilities. The range of SVM output scores is $[-a, b]$, where a and b depend on the data. In LSFS, the most important task is comparing classifiers in order to find the best test. However, we

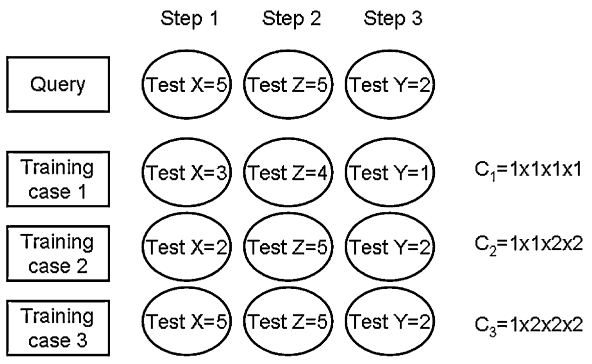

Fig. 3. Instance weight update example.

cannot compare classifiers that are trained with different subsets by raw SVM scores, which are not probabilities. Instead, we need to compare classifiers using calibrated predicted probabilities.

A forecaster is well-calibrated if the predicted probability is close to the observed probability [39,40]. The quality of calibration can be analyzed by a reliability diagram [39], in which the prediction space is separated into 10 bins. Cases with values between 0 and 0.1 are located in the first bin, and cases with the values 0.1 to 0.2 are located in the second bin, etc. Then we calculate and plot the point of the proportion of positive cases against the average predicted probability for each bin. If a forecaster is well calibrated, all points should be close to the diagonal.

The scores of SVM tend to be distributed away from the extremes, and the predicted points would generally form a sigmoid shape on the reliability diagram [41]. A good calibration method can adjust the sigmoid shape to a near-diagonal line on the diagram. Platt [42] used a sigmoid function to calibrate SVM scores. The raw SVM scores are transformed into posterior probabilities using the equation

$$P(y = +1|d) = \frac{1}{1 + \exp(Ad(x) + B)}, \tag{3}$$

where A and B are parameters trained from the data (see Platt [42]).

3.3. Locally sequential feature selection strategies

LSFS selects features for only one case (the query). We use LSFS to guess the next test to speed up diagnosis, so that only tests whose order needs to be determined are considered as features in LSFS (e.g., delayed tests). The degree of importance of a feature may change with differences in known patient information, demographic data, history, physical findings, and previously selected tests results.

One important property of LSFS is the sequence of features (tests). The symptoms and demographic data of each patient varies, so the timing of the same test for two patients may differ. For example, when the predicted probability of disease is very close to either 0 or 1 but has not crossed either threshold, most tests can help to cross the treatment (or no-test) threshold easily. However, if the prediction is close to 0.5, a very strong predictor may be necessary. The next test of these two cases can be very different. As a result, each case will have an unique test sequence because the known information of each patient varies.

There are two selection strategies used with LSFS: speed-based and cost-based. The test selection strategy is represented by the evaluation function f. The selected feature can provide the most information for the query. For speed-based strategies, information means the speed of the next prediction moving toward a predicted probability of either 0 or 1. For cost-based strategies, information means the most cost-efficient choice, considering not only speed but also the cost of a test.

We consider four evaluation functions in each category of selection strategy. For the speed-based category, these evaluation functions are probability contribution, minimum uncertainty, expected uncertainty, and instance weight-expected uncertainty (IWEU). Each function has a corresponding cost-based evaluation function.

We begin by describing the speed-based approaches. When selecting a test, we have to consider which test is most likely to lead to diagnostic certainty (with a post-test probability approaching either 0 or 1). We cannot know the actual value of a test that has yet to be performed, but we can compute the predicted probability of diseases associated with all possible values of that test. We can then find the most promising test that can approach either end the fastest.

The probability contribution (PC) function finds the test that produces the greatest changes in predicted probability of disease.

For all possible values of test i, f is the maximum difference of probabilities. This function is defined as follows:

$$f_{\mathrm{PC}}(\bar{v}\cup u_i)=\max(h(d(\bar{v}\cup u_i^k))-h(d(\bar{v}\cup u_i^{k'}))),\ \forall k,k'\in\{1,\ldots,|u_i|\},k\neq k',$$

where $|u_i|$ is the number of possible values for the feature u_i, $\bar{v}$ is the known information of the query including symptoms, demographic data, and previously selected test results. u_i^k is the value k of test i (unknown information), k is the index of values of the test u_i. Once the result of the selected test is known, it becomes known information. d is the predictive function that returns SVM scores, and h transform SVM scores into probabilities.

The idea of minimum uncertainty (MU) is to find the test that would push the prediction closest to either 0 or 1. We define uncertainty as simply the minimum difference between the predicted probability and zero or one. This function is defined as

$$f_{\mathrm{MU}}(\bar{v}\cup u_i)=\min(0.5-|h(d(\bar{v}\cup u_i^k))-0.5|),\ \forall k\in\{1,\ldots,|u_i|\}.$$

Compared to minimum uncertainty, the expected uncertainty (EU) is a more stable strategy. The uncertainty of each value of a test is weighted by its frequency in the training set. We find the test with minimum f. The expected uncertainty is defined as

$$f_{\mathrm{EU}}(\bar{v}\cup u_i)=\sum_k\frac{(0.5-|h(d(\bar{v}\cup u_i^k))-0.5|)\times Freq^k}{\sum_k Freq^k},$$

where $Freq^k$ is the frequency of training cases with the value u_i^k.

For the IWEU, we use instance weights to replace frequency. The uncertainty of value u_i^k is weighted by the instance weights based on previous tests. Similar to expected uncertainty, we find the test with the minimum f. The formulation can be expressed as

$$f_{\mathrm{IWEU}}(\bar{v}\cup u_i)=\sum_k\frac{(0.5-|h(d(\bar{v}\cup u_i^k))-0.5|)\times C^k}{\sum_k C^k},$$

where C^k is the sum of instance weights of training cases with the value u_i^k.

The last three functions find the test with the smallest uncertainty based on a single (minimum uncertainty) or average (expected uncertainty and IW-expected uncertainty) test result(s). Locations of predictions provide information to guess which test is most likely to move the probability of disease upward or downward. The first function (f_{MU}) finds the test with some value that has the global minimum uncertainty, and the last two functions use expected uncertainty to select a test.

In the cost-based strategy, cost is defined as test cost while effectiveness is the degree of movement toward either end (as in the speed-based strategy). We use the ratio *effectiveness/cost* instead of only *effectiveness*.

Each speed-based evaluation function has a corresponding cost-based function. The cost-based version of probability contribution becomes probability contribution per dollar. We want to select the test with maximum f. The cost-based objective function is

$$f_{\mathrm{costPC}}(\overline{v}\cup u_i)=\frac{\max(h(d(\overline{v}\cup u_i^k))-h(d(\overline{v}\cup u_i^{k'})))}{\mathrm{cost}_i},\quad \forall k,k'\in\{1,\ldots,|u_i|\},k\neq k',$$

where cost_i is the cost of using test i.

All cost-based functions in the uncertainty family become uncertainty reduction per dollar. For uncertainty reduction, we need to compute the reduction of uncertainty from known information $\bar{v}$. The uncertainty of known information is defined as $UC(\bar{v})=(0.5-|h(d(\bar{v}))-0.5|)$. Cost-based minimum uncertainty reduction is

$$f_{\mathrm{costMU}}(\bar{v}\cup u_i)=\frac{UC(\bar{v})-(0.5-|h(d(\bar{v}\cup u_i^k))-0.5|)}{\mathrm{cost}_i},\quad \forall k\in\{1,\ldots,|u_i|\}$$

We select the test with maximum f because we want to select the test with maximum uncertainty reduction per dollar. Cost-based expected uncertainty reduction is

$$f_{\mathrm{costEU}}(\bar{v}\cup u_i)=\frac{UC(\bar{v})-(\sum_k(0.5-|h(d(\bar{v}\cup u_i^k))-0.5|)\times Freq^k)/\sum_k Freq^k}{\mathrm{cost}_i},$$

and we want to find a test with the maximum f.

Similarly, cost-based IW expected uncertainty reduction is

$$f_{\mathrm{costIWEU}}(\bar{v}\cup u_i)=\frac{UC(\bar{v})-(\sum_k(0.5-|h(d(\bar{v}\cup u_i^k))-0.5|)\times C^k)/\sum_k C^k}{\mathrm{cost}_i}.$$

3.4. The ODPF algorithm

Algorithm 1 summarizes the whole process. The input dataset D consists of patients' known information V, such as symptoms and known results of tests, and delayed tests (or unknown information) U. Initially, this study assigns all immediate tests to V based on the definition of Turney [30]. When the result of U_j is known, this test becomes known information. θ is a confidence threshold in the range of 0 to 0.5. Thus, $1-\theta$ represents a treatment threshold and θ represents a no-test threshold. The limitations of the $[\theta, 1-\theta]$ threshold structure, along with a remedy, is discussed in Section 3.7.

Algorithm 1 (ODPF).

```
Input:
    1 Data $D$, $D = V \cup U_j$, $j \in \{tests\}$, $k \in$ data points
    2 Threshold $\theta$ and weight update $\gamma$
    3 Query $\bar{v}$
Output:
    1 Test sequence $SEQ$ before $\theta$ is met
    2 Prediction
1: C = 1
2: $UT = \{\text{tests}\}$
3: $SEQ = \varnothing$
4: for $j = 1 \ldots |tests|$ do
5:     $[\hat{i}, h, d] = ChooseClassifier(D, \bar{v}, C, UT)$
6:     $UT = UT/\hat{i}$ and $U = U/U_{\hat{i}}$
7:     $\bar{v} = \bar{v} \cup \bar{u}_{\hat{i}}$ and $V = V \cup U_{\hat{i}}$
8:     $IW_k = IW_k * \gamma,\ \forall k : u_{k\hat{i}} = \bar{u}_{\hat{i}}$
9:     $SEQ = SEQ\ \&\ \hat{i}$
10:    if $h(d(\bar{v})) < \theta$ or $1 - h(d(\bar{v})) < \theta$ then
11:        return $[h(d(\bar{v})), SEQ]$
12:    end if
13: end for
14: return $[h(d(\bar{v})), SEQ]$
```

```
ChooseClassifier $(D, \bar{v}, C, UT)$:
15: $\hat{i} = 1$
16: for $i = 1 \ldots |UT|$ do
17:     $d = GetPredictor(V \cup U_i, C)$
18:     $h = GetCalibrationFunction(V \cup U_i)$
19:     if $f(\bar{v} \cup \bar{u}_i) > f(\bar{v} \cup \bar{u}_{\hat{i}})$ then
20:         $\hat{i} = i, h^* = h, d^* = d$
21:     end if
22: end for
23: return $\hat{i}, h^*, d^*$
```

The query with known information $\bar{v}$ activates the recommendation process. We want the algorithm to recommend a test at each step i after obtaining the previous test result $\bar{u}_{\hat{i}-1}$, and decide when to stop. At termination, the output is the diagnostic decision and the sequence of previously recommended tests.

In step 1, we set the instance weight (C) of each case to be the same. Step 2 initializes the pool of unused tests (UT) to the set of all tests, and step 3 initializes the test sequence (SEQ).

Steps 4–13 are the repeating processes for confident diagnosis and test selection. Step 5 computes the best test $\hat{i}$, the corresponding calibration function h (Eq. (3)), and the decision function d (Eq. (2)). Step 6 removes $\hat{i}$ from UT and removes $U_{\hat{i}}$ from U. After the test value of $\hat{i}$ is revealed, step 7 adds its value $\bar{u}_{\hat{i}}$ to $\bar{v}$ because the test result $\bar{u}_{\hat{i}}$ has become known information for the query case. Also, the column of training features $U_{\hat{i}}$ will be added to V. We do not add or remove data from the dataset D, but, in each iteration, we move a certain feature from U to V. Step 8 updates instance weights. When the result of the selected test for a training case k matches the query case, the instance weight of k is updated. Step 9 appends test $\hat{i}$ to SEQ.

Steps 10–12 decide whether the prediction is confident enough. If the answer is positive, the algorithm will return the diagnosis and the test sequence. Otherwise, the process repeats until all tests have been run.

For the test searching subroutine, step 15 assigns the first test as the selected one. Then, steps 16–22 update the selection. In each iteration, a trial dataset, which consists of training features (V, corresponding to the known information of the query) and an unused test feature (U_i), is used to train a decision function d (step 17) and a calibration function h (step 18).

Steps 19–21 use a function f to decide the best test. When test i is better than test $\hat{i}$, we record d^* and h^* of the new $\hat{i}$. Step 23 returns $\hat{i}$, d^*, and h^* of the best test.

3.5. Speed-up strategy: inheritance

The ODPF algorithm uses a large number of classifiers to select the best test (or sequence of tests). Training these classifiers is computationally expensive. To reduce this problem, we allow a query to be able to inherit classifiers trained from a previous query. A new query case can always share the first group of classifiers to identify the first test because C is 1 for all cases. After selecting the first test, if the test result of the query is the same as a previous query, the query can inherit classifiers from that previous query. Fig. 4 illustrates the rule of inheritance. Unlike decision tree algorithms, a node represents a group of classifiers that determine the next test; one can also make a decision at this node. Instead of explicit rules, the choice of test is driven by classifiers. Query 1 has to train all classifiers for identifying tests. Test 5 is the first selected test. After performing this test, we obtain the value 1 for test 5. Next, test 6 is suggested by the second group of classifiers.

Query 2 can inherit the first group of classifiers. Test 5 was again selected but the value is −1. Query 2 is ineligible to inherit any more because lazy classifiers are trained specifically for a query.

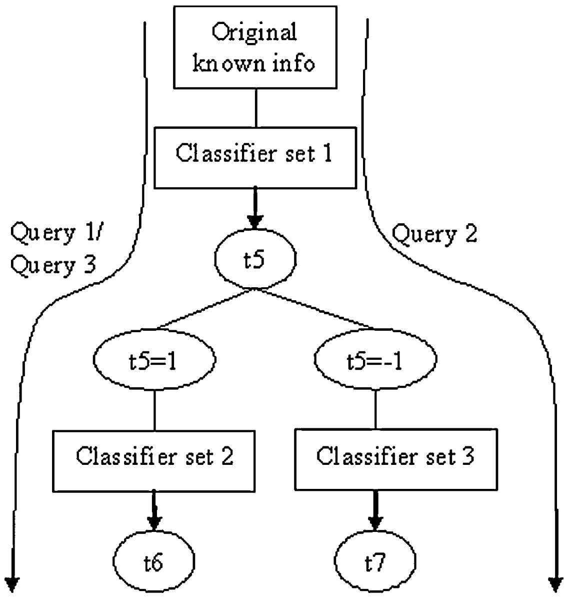

Fig. 4. The rules of inheritance.

Thus, a query (e.g., query 2) can inherit a classifier set only when all previous tests and their values are exactly the same as a previous query (e.g., query 1). Therefore, we have to train a new group of classifiers (classifier set 3) for query 2 in order to identify the next test. For query 3, not only is test 5 the first selected test but also the value is the same as query 1. Thus, query 3 can inherit both groups of classifiers from query 1.

3.6. Missing values

Missing values are common in medical datasets, especially those recording diagnostic tests, because a physician will not run tests considered to be unnecessary. Therefore, ODPF must adjust for missing values in both training and validation. First, for training, we impute a missing value using the class-conditional mean of the feature (i.e., the mean feature value of all points within the same class as the instance with the missing value).

Second, the C update depends on the proportion of the matching value in the training set. For example, consider a test with three possible values, 1, 2, and 3, whose proportions are 30%, 30%, and 40% in the training set. If the new test value of the query is 3, for example, each training instance with a missing value of this feature can be updated, but the instance-weight multiplier becomes $1 + (\gamma - 1) \times 40\,\%$.

When we apply ODPF in an actual clinical setting, all test values of a query (a new patient) are supposed to be missing, and we do not have any problem for the query with missing values. However, when validating ODPF, we need to slightly modify the query case with some missing values in order to validate reasonably. When some test values of a query are missing, we limit the selection of tests to those without missing values. When a diagnosis is highly uncertain, all available tests (without missing values) may not allow one to cross a probability threshold. In this case, the diagnosis must be made after receiving values of these available tests.

3.7. Unbalanced data adjustment

In some situations (e.g., a rare disease), the separation surface of the trained SVM will not correspond to a probability of 0.5. Therefore, unequal thresholds, such as $[\theta_1, \theta_2]$, may be better than equal thresholds $[\theta, 1–\theta]$.

The definition of uncertainty also needs to change in an unbalanced dataset. When a dataset is balanced, uncertainty is determined using either 0 or 1 as a reference. In an unbalanced dataset, uncertainty is decided using two given thresholds, instead of a single offset from 0 to 1. In an unbalanced dataset, we are still looking for a test with the smallest uncertainty in minimum uncertainty and (IW-)expected uncertainty functions. The use of asymmetrical $[\theta_1, \theta_2]$ thresholds may be desired in situations beyond the need to balance a skewed class distribution. For example, one may desire a very high diagnostic confidence level before initiating a treatment with significant side effects.

In the original defintion of uncertainty, the smallest uncertainty is 0. In the new definition, when a test with some set of values crosses the threshold, its computed value of uncertainty may be less than 0. In this case, one can still identify the test with the lowest value of uncertainty, and we still can use all speed-based and cost-based test selection functions as described in Section 3.3.

3.8. Multi-class strategies

Many diagnosis problems involve a choice among several candidate conditions instead of a single disease. We show that the ODPF algorithm can be adjusted for multi-class decision problems. Similar to binary problems, we wish to find a test that can approach one possible outcome (class) most quickly. The difference is that we are comparing several possible outcomes instead of two, but we can still use the test-selection functions described in Section 3.3. For example, in a binary problem, we are finding the test with the smallest uncertainty from either end (probability 0 or 1, two possible outcomes). In a multi-class problem, we are finding the test with the smallest uncertainty for one of the outcomes.

There are several algorithms for solving multi-class problems [43] using SVMs. We use the adapted one-against-all method as the base classifiers. Assuming n possible classes (candidate conditions), we need to train $n - 1$ classifiers with $n - 1$ sets of labels. Each set of labels indicates whether or not a record belongs to a specific class. Each class corresponds to a set of labels except for one class.[1] If the decision function of only one class is greater than zero, this class is the predicted class. If the decision function of all classes are less than zero, the majority class is the predicted class. When there is more than one class with decision function greater than zero, the predicted class is the class with the highest predicted probability.

3.9. Data sources

3.9.1. Datasets for diagnosis

We apply the ODPF algorithm to heart disease and thyroid datasets. Both datasets are obtained from the UCI machine learning repository [44].

Heart disease dataset [45]: This analysis sample included 303 consecutive patients (mean age 54 years, 68% male) who were referred for coronary angiography at the Cleveland Clinic between May 1981 and September 1984. No patient had a history of prior myocardial infarction or known valvular disease or cardiomyopathy. All patients received a history and physical examination, resting electrocardiogram, and laboratory studies as part of their routine evaluation. Study patients also underwent exercise stress testing, myocardial perfusion imaging, and cardiac fluoroscopy as part of a research protocol. The dataset included 4 clinical variables (age, sex, chest pain type, and systolic blood pressure), 2 laboratory variables (serum cholesterol and fasting glucose), and resting electrocardiographic variables (ST segment depression >0.05 mV or T-wave inversions, probable or definite left ventricular hypertrophy based on Estes' criteria). The diagnosis of coronary artery disease was defined by the presence of $>50\%$ narrowing of one or more coronary arteries on angiography. All coronary angiograms were interpreted by a cardiologist who was blinded to non-invasive test results.

The heart disease dataset came from Cleveland Clinic Foundation and was provided by the principal investigator Robert Detrano of the V.A. Medical Center, Long Beach and Cleveland Clinic Foundation. In this experiment, the dataset was downloaded from the Statlog project; 6 data points were discarded because of missing class values and 27 data points were retained in case of dispute [46], at the UCI machine learning repository. Originally, there were four different types of disease in this dataset. In this project, the classification task is simply to distinguish the presence of disease (all four types) from its absence. The dataset does not have missing values. The proportion of patients with disease in the dataset is 44.4%.

Thyroid disease dataset [47]: This analysis sample consists of 3772 cases which were referred to the Garvan Institute of St. Vincent's Hospital, Sydney, Australia for diagnostic evaluation starting in late 1984. The dataset includes 11 clinical attributes, which were abstracted from data provided by the referring physician (age, sex, pregnancy, goiter, tumor, hypopituitarism, use of thyroid or antithyroid medication, history of thyroid surgery, complaint of malaise, psychological symptoms), and up to six test results (including TSH, T3, TT4, T4U, free thyroxine index, and TBG) requested by the referring physician. Final diagnosis of hyperthyroidism, hypothyroidism, or euthyroid status was based on interpretation of all available clinical and laboratory data by a qualified endocrinologist (or in some cases, an expert system designed to diagnose thyroid disease). We follow Turney's paper [30] using four tests (TSH, T3, TT4, and T4U) because costs are provided for only those tests. The resulting dataset does not include missing values. The proportion of cases with hypothyroidism and hyperthyroidism is 2.5 and 5.1%, respectively.

3.9.2. Datasets for prediction of future risk

We have also performed an analysis of two additional datasets in order to show the potential applicability of the ODPF method in predicting the future risk of disease or adverse events with fewer tests by determining an optimum patient-specific sequence. Both datasets are also obtained from the UCI machine learning repository [44]. These datasets are briefly described below:

Pima Indians diabetes dataset [48]: The dataset was collected by the National Institute of Diabetes and Digestive and Kidney Diseases. All subjects were females at least 21 years old of Pima Indian heritage. The dataset includes 6 clinical variables (age, diabetes pedigree function, body mass index, triceps skin fold thickness, diastolic blood pressure, and number of pregnancies) and two tests (glucose tolerance test and serum insulin test) to classify the risk of diabetes. The proportion of subjects who developed diabetes is 34.9%.

Hepatitis dataset [49]: The dataset includes 14 clinical attributes (age, sex, patient on steroids, antiviral therapy, fatigue, malaise, anorexia, hepatomegaly, palpable firmness of liver, palpable spleen, presence of spider veins, ascites, presence of varices, and liver histology), laboratory tests (bilirubin, alkaline phosphotase, aspartate aminotransferase, albumin, and protime), and the prognostic outcome of disease (live or die). The proportion of subjects who died during follow-up is 20.7%.

4. Results

In this section, we show results in heart disease, thyroid, diabetes, and hepatitis datasets. All features are discretized based on the Fayyad and Irani method [50] implemented in ROSE [51]. All

[1] In this project, we choose the majority class.

results were generated from RBF-SVM and averaged over five 10-fold cross-validation (CV) runs. We use the 10-fold CV to replace leave-one-out [52] in order to benefit from inheritance as described in Section 3.5. In other words, each data point is a query that can share classifiers with another query in the same testing fold. The probability transformation function (Eq. (3)) was trained in 3-fold CV runs.

We used over-sampling of the minority class [53] to balance each dataset according to the proportion of positive to negative cases. For example, for a dataset with 95% positive cases, we used the ratio 1/19 to balance the positive and negative classes.

In our cost analyses, we apply a group discount when other tests belonging to the same group have already been requested. For example, blood-related tests have a group price because they share the cost of phlebotomy.

We used $\gamma = 1.4$ for constructing lazy SVM classifiers. The parameter was determined by predictive performance. For instance, the total accuracies of $\gamma = 1, 1.4, 1.8$ in Probability Contribution and minimum Uncertainty functions are [0.837, 0.8407, 0.8185] and [0.8556, 0.8593, 0.8333], respectively. SVM classifiers turn into lazy SVM classifiers when γ is greater than 1. With an appropriate γ, the predictive performance of lazy learning can be improved.

We discuss our results in detail using the heart disease dataset. We demonstrate the multi-class result on the thyroid dataset with the minimum uncertainty function and report aggregated results of diabetes and hepatitis using all LSFS functions.

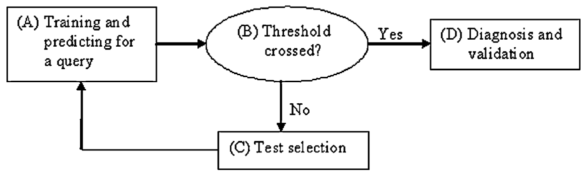
(A) Training and predicting for a query
(B) Threshold crossed?
Yes
(D) Diagnosis and validation
No
(C) Test selection

Fig. 5. Evaluation process.

Our baseline (or "standard") in each case is a classifier trained with all available tests from from the dataset. ODPF aims to identify the minimum set of tests. We show the comparison of the number of tests, costs, and accuracies between all functions and baseline in the following subsections.

4.1. Heart disease dataset

Fig. 5 summarizes the process to evaluate the model. We train and predict for each query patient (A). If a prediction does not cross either the treatment or no-test threshold, more testing is needed (from B to C). We can utilize several different test selection functions in C, e.g., minimum uncertainty. One test will be appended to the sequence each time once goes through the A,B,C,A cycle. If the prediction crosses either threshold, a diagnosis is made (D) and validated.

Tables 1 and 2 summarize the diagnosis accuracy and cost savings of four evaluation functions. The notation ite1 to ite9

Table 1
Speed-based ODPF. The treatment and no-test threshold is [0.85, 0.15]. The accuracy of the baseline that trained with 9 tests is 0.8407. + or − indicates that ODPF is significantly better or worse than baseline, respectively ($\alpha = 5\%$).

Functions	Statistics	ite0	ite1	ite2	ite3	ite4	ite5	ite6	ite7	ite8	ite9
ProbContr	Correct	238	146	77	109	127	71	45	50	44	219
	Total	265	180	83	120	152	82	50	62	51	305
	Accuracy	0.89811	0.81111	0.92771	0.90833	0.83553	0.86585	0.9	0.80645	0.86275	0.71803
Total accuracy = 0.834; average test = 4.16$^+$; cost savings = 53.13$^+$%											
minUC	Correct	238	374	146	129	36	34	16	17	13	147
	Total	265	416	157	153	47	38	26	23	13	212
	Accuracy	0.898	0.899	0.930	0.843	0.766	0.895	0.615	0.739	1	0.693
Total accuracy = 0.852$^+$; average test = 2.89$^+$; cost savings = 49.59$^+$%											
expUC	Correct	238	186	218	97	77	55	37	26	20	179
	Total	265	220	232	111	97	62	50	32	23	258
	Accuracy	0.898	0.845	0.940	0.874	0.794	0.887	0.740	0.813	0.870	0.694
Total accuracy = 0.8393; average test = 3.51$^+$; cost savings = 51.52$^+$%											
IWexpUC	Correct	238	188	210	100	79	53	37	30	19	178
	Total	265	220	223	114	100	62	51	37	21	257
	Accuracy	0.898	0.855	0.942	0.877	0.790	0.855	0.725	0.811	0.905	0.693
Total accuracy = 0.839; average test = 3.53$^+$; cost savings = 51.23$^+$%											

Table 2
Cost-based ODPF. The treatment and no-test threshold is [0.85, 0.15]. The accuracy of the baseline that trained with 9 tests is 0.8407. + or − indicates that ODPF is significantly better or worse than baseline, respectively ($\alpha = 5\%$).

Functions	Statistics	ite0	ite1	ite2	ite3	ite4	ite5	ite6	ite7	ite8	ite9
ProbContr	Correct	238	89	70	104	152	68	67	64	44	226
	Total	264	111	79	113	181	77	74	80	55	316
	Accuracy	0.902	0.802	0.886	0.920	0.840	0.883	0.905	0.800	0.800	0.715
Total accuracy = 0.831; average test = 4.45$^+$; cost savings = 53.38$^+$%											
minUC	Correct	238	35	82	71	203	73	53	83	21	258
	Total	265	43	93	77	223	90	62	118	24	355
	Accuracy	0.898	0.814	0.882	0.922	0.910	0.811	0.855	0.703	0.875	0.727
Total accuracy = 0.827$^-$; average test = 4.7311$^+$; cost savings = 54.09$^+$%											
expUC	Correct	238	59	84	88	218	70	79	40	32	214
	Total	265	66	101	106	230	88	105	49	40	300
	Accuracy	0.898	0.894	0.832	0.830	0.948	0.795	0.752	0.816	0.800	0.713
Total accuracy = 0.831; average test = 4.40$^+$; cost savings = 54.97$^+$%											
IWexpUC	Correct	238	57	81	92	214	68	78	42	35	216
	Total	265	67	93	110	227	86	106	49	43	304
	Accuracy	0.898	0.851	0.871	0.836	0.943	0.791	0.736	0.857	0.814	0.711
Total accuracy = 0.830; average test = 4.43$^+$; cost savings = 54.64$^+$%											

represents the number of delayed tests obtained by patients. Ite1 represents only 1 test performed, while ite4 represents 4 tests performed. Ite0 is the situation where a diagnosis (prediction) is made without obtaining any delayed tests. The sequence of tests of patients may vary, so we cannot show actual test sequences (e.g. [test5, test3, test1, . . .]) of all patients in these aggregated results. Results in these two tables are predictive results at different stopping points in the test sequence.

The stopping point for patients varies because the number of tests needed to diagnose is based on the individual patient. Some patients need few tests while others need many to support a diagnostic decision. Thus, patients may stop with any number of tests, and we summarized the aggregated results in ten strata (ite0 to ite9). When testing is stopped, we add one to the Total row, and compare the prediction (or diagnosis) with its class label (which shows whether the diagnosis is correct or not) at this stopping point. If the prediction is correct, one will be added to the Correct row. The percentage of correct predictions is shown in the Accuracy row. Total accuracy is the expected accuracy of all strata. "Average test" is the average number of tests used, and cost savings is the average cost saving of unused tests for all patients.

These two tables also show the accuracy of diagnosis in each iteration. Some patients do not need to receive any testing (ite0), but other patients need to receive all tests (ite9). The groups in ite0 to ite9 correspond to patients with increasing difficulty of diagnosis. In other words, when a case stops early, its predictive probability is far away from the separation surface and satisfies the stopping criterion. On the other hand, the patients in ite9 remain close to the separation surface. Thus, the accuracies for early diagnosis patients tend to be high (ite0 to ite8).

For patients who remain in the ite9 group, one still needs to make a diagnosis for these patients once all available tests have been exhausted. As these patients are difficult to diagnose, the diagnostic accuracy for the ite9 group is low. Interestingly, the threshold plays the role of a filter by keeping patients in the appropriate stratum.

Both tables use + and − to indicate whether a number is significantly better or worse than the baseline, in which all tests were performed ($\alpha = 5\%$). The accuracy of the baseline is 0.8407. Although total accuracies of most functions are lower than baseline, the differences except for the cost-based minimum uncertainty (minUC) function are not significant. However, the accuracy of minUC function (0.852) is significantly better than the baseline. Average number of tests and cost savings of all functions are significantly better than the baseline. The number of tests required ranged from 2.89 to 4.73 while most cost savings are more than 50%. In general, speed-based strategies diagnose with fewer tests than cost-based strategies, and cost-based strategies save more cost than speed-based strategies.

Fig. 6 compares accuracies of three thresholds that use the minimum uncertainty strategy. The x-axis represents the stopping points. Standard refers to the baseline classifier that is trained with all tests. A larger threshold has greater accuracy in all strata. However, total accuracies for thresholds 0.75, 0.85, and 0.95 are 0.813, 0.852, and 0.847, respectively. In other words, a higher threshold does not always result in higher total accuracy. This is attribute to the effect of early diagnosis.

Fig. 7 summarizes the frequency of testing across the three thresholds. Many patients can be diagnosed early (i.e., before exhausting all tests). As expected, the barrier of a higher threshold results in fewer patients being diagnosed early. Therefore, more patients remained until ite9 when the threshold is higher. Total accuracy is an expected accuracy which results from the combination of accuracy and frequency in each stratum. The accuracies of early diagnostic strata (ite0 to ite8) are generally high, but the accuracy for ite9 is low. A very high threshold such as

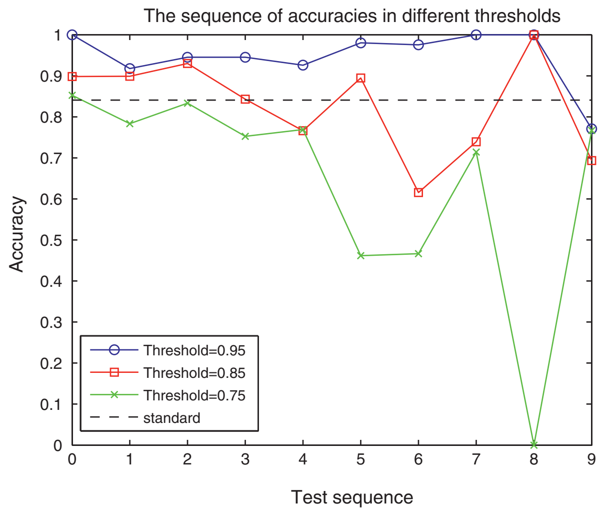

Fig. 6. Accuracies of various thresholds.

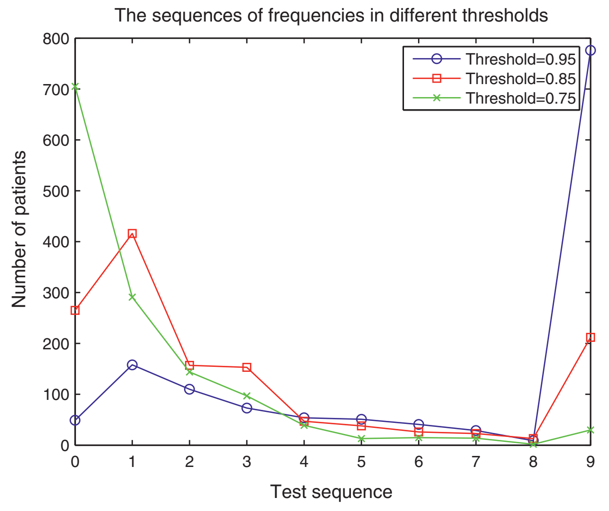

Fig. 7. Frequency of each test.

0.95 forces many patients to stay until ite9 and largely reduces the total accuracy. Although accuracies of most strata of an overly high threshold, such as 0.95, are higher, overall reduction (ite9) in the number of patients who are assigned a diagnosis outweighs this improvement (ite0 to ite8). Therefore, careful selection of an appropriate threshold helps to improve total accuracy. The fluctuation of accuracies in Fig. 6 also results from early diagnosis, which results in small sample size from strata ite4 to ite8.

The relationship between effectiveness and costs is shown in Fig. 8. We compare random, minimum uncertainty, and cost-based expected uncertainty. Random is the baseline strategy in which testing sequences are randomly created. This baseline allows the examination of the amount of contribution due to feature-selection strategies (i.e., minimum uncertainty and cost-based expected uncertainty). Random still benefits from the confident prediction structure because the threshold still determines the stopping point. Therefore, many tests can still be saved. For example, when the threshold is 0.85, the system uses 3.68 tests with a total accuracy of 0.827.

In Fig. 8, we demonstrate that cost savings and probability threshold are negatively related. As expected, both strategies have more cost savings and higher total accuracies than random. In general, cost savings and total accuracy are a trade-off. Surprisingly, the minUC strategy provides more cost savings, especially

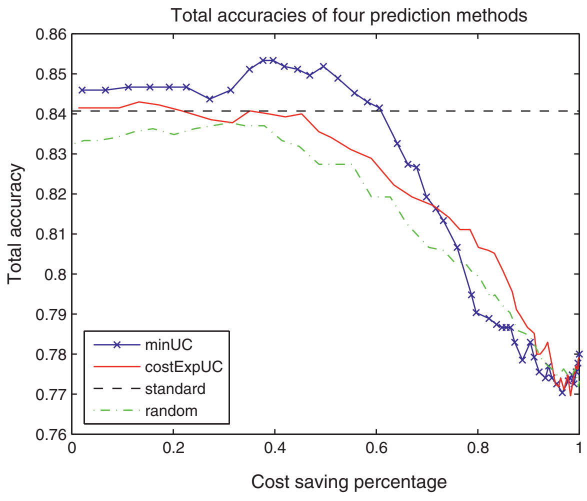

Fig. 8. Accuracy vs. cost savings.

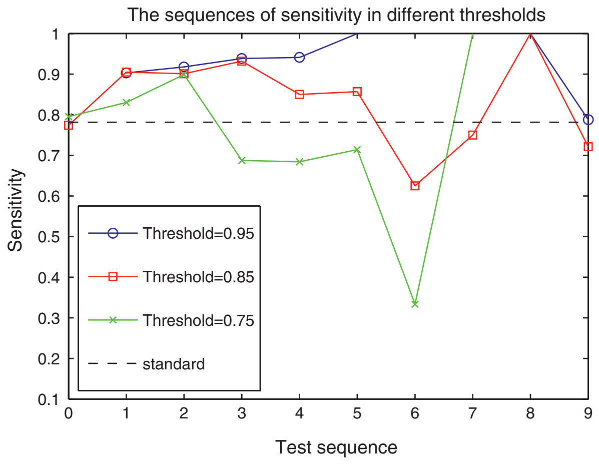

Fig. 9. Sensitivity of each test.

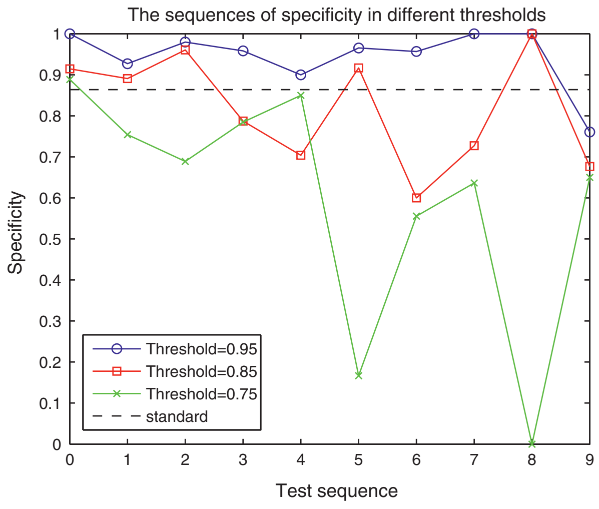

Fig. 10. Specificity of each test.

when the threshold is higher. For example, when the cost savings is 50%, the accuracy for minUC is 85.1% while the accuracy for costExpUC is 83.4%. The cost savings of the speed-based strategies comes from using a small number of tests that are usually expensive but strong predictors.[2] However, the cost savings of the cost-based strategies results from careful selection of tests. The selection prefers inexpensive tests with acceptable diagnostic power or powerful tests with reasonable cost. This result suggests that using expensive tests at the outset may be preferable in some situations. When the sequences are optimized, patients may be diagnosed more quickly and accurately, and more cost can be saved.

Confident prediction not only improves accuracy but also sensitivity and specificity. Figs. 9 and 10 show sensitivity and specificity of each stratum for the minUC strategy. Both figures show that sensitivity and specificity for higher thresholds are better. Similar to accuracy, early diagnosis results in fluctuation of the curve. Standard represents sensitivity or specificity of the classifier trained with all tests. Similar to accuracy, confident prediction boosts both sensitivity and specificity for patients who are diagnosed early.

4.2. Thyroid dataset

The thyroid dataset is highly unbalanced and has three classes, normal (92.5%), hyperthyroid (5.1%), and hypothyroid (2.5%). Tables 3 and 4 show confusion matrices of thyroid dataset from baseline (the classifier trained with all features) and ODPF using the function of minimum uncertainty.

For both tables, the third to fifth columns show predicted class frequencies of hypothyroid, hyperthyroid, and normal. The last column, matched rate, is the proportion of correct predictions among all predictions in the same true class. For example, in Table 3, the matched rate of hypothyroidism is 250/465 = 0.538. Similarly, the third to fifth rows show true class frequencies. The predicted-matched rate is the proportion of correctness of a predicted class among all true classes in the same prediction. For example, the predicted-matched rate of hypothyroid is 250/389 = 0.643. Total accuracy is the percentage of correctly classifies cases.

Comparing these two tables, most predicted performance indices are better than the baseline. The average number of tests and total cost of baseline are 4 and 53.81 (Table 3). In this dataset, we force the algorithm to give at least one test to all patients to improve prediction. On average, ODPF uses 1.07 tests and the total cost is 23.37 (cost savings 56.6%), both of which are significantly better than the baseline.

4.3. Application of the ODPF method in predicting future risk of disease and adverse events

To determine whether ODPF can be extended to applications of diagnostic tests for prognosis and risk stratification, we performed an analysis of diabetes and hepatitis datasets.

Tables 5 and 6 show the aggregated results of all strata for the diabetes and hepatitis datasets. For the purpose of demonstration, we use [0.75, 0.1] and [0.75, 0.08] as treatment and no-test thresholds for diabetes and hepatitis datasets, respectively. The first column of all tables shows all functions including speed-based methods and cost-based methods, and the following columns are accuracy, sensitivity, specificity, area under the ROC curve (AUC), cost, and average number of tests.

Standard is the baseline classifier that was trained with all features. Only the hepatitis dataset has missing values. Missing values are imputed using the class-conditional mean of a feature. Average tests and costs are computed based on tests and costs actually used and spent. Thus, the average number of tests in the

[2] In most cases, the selection starts from expensive and powerful tests. In some cases, an inexpensive test is preferred in the beginning depending on the known information before taking any tests.

Table 3
Confusion matrix of the baseline.

		Predicted class			
		Hypothyroid	Hyperthyroid	Normal	Matched rate
True class	Hypothyroid	250	200	15	0.538
	Hyperthyroid	54	681	220	0.713
	Normal	85	108	17,247	0.989
	Predicted matched rate	0.643	0.689	0.987	
Total accuracy: 0.964; cost: 53.81 (0% saving); average number of tests: 4					

Table 4
Total confusion matrix of ODPF (minimum uncertainty). The treatment and no-test thresholds for both hyperthyroid and hypothyroid are [0.85, 0.02]. + or − indicates that ODPF is significant better or worse than baseline, respectively ($\alpha = 5\%$).

		Predicted class			
		Hypothyroid	Hyperthyroid	Normal	Matched rate
True class	Hypothyroid	352	99	14	0.7570^+
	Hyperthyroid	78	818	59	0.8565^+
	Normal	136	108	17,196	0.9860^-
	Predicted matched rate	0.622^-	0.798^+	0.996^+	
Total accuracy: 0.974^+; cost: 23.37 ($56.6\%^+$ saving); average number of tests: 1.07^+					

Table 5
Diabetes summary. The treatment and no-test threshold is [0.75, 0.1]. + or − indicates that ODPF is significant better or worse than baseline, respectively ($\alpha = 5\%$).

Functions	Accuracy	Sensitivity	Specificity	AUC	Cost (save%)	Average tests
Standard	0.735	0.721	0.743	0.828	38.29 (0)	2
ProbContr	0.735	0.728^+	0.739^-	0.827^-	$27.80\ (27.4)^+$	1.474^+
minUC	0.735	0.727^+	0.739^-	0.827^-	$27.81\ (27.4)^+$	1.474^+
expUC	0.735	0.724	0.741^-	0.827	$30.04\ (21.5)^+$	1.578^+
expIWUC	0.735	0.722	0.742^-	0.827^-	$29.97\ (21.7)^+$	1.576^+
ProbContr_Cost	0.735	0.728^+	0.738^-	0.827^-	$27.82\ (27.3)^+$	1.475^+
minUC_Cost	0.735	0.728^+	0.738^-	0.827^-	$27.76\ (27.5)^+$	1.472^+
expUC_Cost	0.734^-	0.722	0.741^-	0.827	$29.98\ (21.7)^+$	1.576^+
expIWUC_Cost	0.734^-	0.722	0.741^-	0.828	$30.00\ (21.7)^+$	1.577^+

Table 6
Hepatitis summary. The treatment and no-test threshold is [0.75, 0.08]. + or − indicates that ODPF is significant better or worse than baseline, respectively ($\alpha = 5\%$).

Functions	Accuracy	Sensitivity	Specificity	AUC	Cost (save%)	Average tests
Standard	0.819	0.744	0.839	0.863	24.78 (0)	4.213
ProbContr	0.817	0.675^-	0.854^+	0.857	$16.37\ (33.9)^+$	2.688^+
minUC	0.817	0.675^-	0.854^+	0.856	$15.96\ (35.6)^+$	2.622^+
expUC	0.819	0.700^-	0.850^+	0.859	$15.94\ (35.7)^+$	2.635^+
expIWUC	0.830^+	0.744	0.852^+	0.859	$14.05\ (43.3)^+$	2.394^+
ProbContr_Cost	0.817	0.675^-	0.854^+	0.857	$16.35\ (34.0)^+$	2.689^+
minUC_Cost	0.817	0.675^-	0.854^+	0.855	$15.90\ (35.9)^+$	2.628^+
expUC_Cost	0.819	0.700^-	0.850^+	0.858	$15.98\ (35.5)^+$	2.645^+
expIWUC_Cost	0.830^+	0.744	0.852^+	0.859	$14.06\ (43.3)^+$	2.395^+

baseline of hepatitis is not an integer (4.213, see Table 6). We do not know the order of tests used in this database. When taking group discount into account, there may be more than one set of costs for a patient, and we take the minimum one for the baseline.

Although only two tests are involved in diagnosing diabetes, not all of them all required. All functions can reduce test cost from 21.5% to 27.5% while using an average number of tests ranging from 1.472 to 1.578. Most accuracies are close to baseline, while cost and average number of tests are all significantly better with the ODPF method compared to baseline.

AUC can be improved because of the feature selection, but AUC can also degrade significantly due to early diagnosis. AUC is computed from predicted probabilities and corresponding labels of patients in all strata. However, treatment (or no-test) thresholds may stop us from improving probabilities. For example, consistent test results usually can further improve predicted probability, but ODPF stops further testing when the current probability is confident enough. Therefore, the limitation of improving predicted probability can degrade AUC, especially when the threshold is not high.

The hepatitis dataset includes many missing values. Although both baseline and ODPF are trained and tested with the same dataset, the uses of the dataset are not exactly equal for two reasons. First, missing values influence the test selection method. In order to train an SVM classifier, we impute missing data. The dataset for the baseline is fully imputed in the training and testing data. In ODPF, we still can impute missing values for training data and update C based on the rule as described in Section 3.6. However, for the query case, we do not impute missing values for the query. Instead, we avoid selecting tests with missing values. In other words, the selection of a test with strong potential to pass the threshold can be prohibited because the test feature value is missing in the query case.

Second, missing values influence the reporting of performance. A test with a missing value for a query case will be moved to the end of the test sequence, and it is marked as N/A because we do not use it. Such a query case will be either early diagnosis or needing more tests than those available. Most cases are in the first situation, and a few of them are in the second. The performance indices of the second situation are reported based on the last test without a missing value. These query cases are forced to attain a diagnosis on account of running out of tests with recorded values. In other words, their diagnoses are not ready but have to be made. The above two reasons may degrade the performance of all functions slightly.

Table 6 shows that specificity, number of tests, and cost savings of all functions of the hepatitis dataset are significantly better than the baseline. In this experiment, both expIWUC and cost-based expIWUC have the highest diagnostic performance and the largest cost reduction among all functions.

Previous results suggest that either sensitivity or specificity can be higher in ODPF. The either-or relationship can be adjusted by tuning γ (see Algorithm 1) or the balance of positive to negative classes based on diseases.

5. Conclusion

This paper presents a novel expert-system construction technique, the ODPF algorithm, which can speed up diagnosis and reduce cost without loss of diagnostic performance (or sometimes, with better performance). Based on our results, cost and the number of diagnostic tests are significantly reduced while predicted performance indices are close to or better than the baseline.

The ODPF algorithm builds upon existing diagnostic methods, such as probability revision and the use of probability thresholds. ODPF allows diagnosis to be more individualized, i.e., individual test sequence construction, the selection of the next test based on pretest probability, the potential effect of the next test, and lazy learning methods.

Three elements work together in ODPF: lazy learning, confident diagnosis, and LSFS. Lazy learners are adapted based on patient results; thus, the predictive models become more individualized when more test results are known. Confident diagnosis determines whether or not to diagnose based on a probability threshold. LSFS is the mechanism for selecting the best test sequence that is most likely to approach the threshold. It can eliminate the need for many tests, as further tests may be unnecessary once the probability of disease crosses the selected threshold. Our method requires sufficient confidence in making a diagnosis before deciding to forgo any diagnostic tests. The total process relies on probabilities of disease before and after taking each test. The choice of the most promising test also relies on examining a range of probabilities associated with possible test results. Thus, one can “guess” the most promising test based on a range of predicted post-test probabilities. In highly uncertain cases, the patient may still receive all tests when no single test is sufficiently powerful to exceed the threshold probability for diagnosis.

ODPF is an ML-based expert system which can avoid the bottleneck of knowledge elicitation. The algorithm can learn clinical knowledge directly from a given dataset with minimum human input. The only knowledge required is the cost of each test, probability thresholds, and the multiplier γ for lazy learning. A suitable γ may result in improvement of both cost and diagnostic performance. An appropriate treatment (or no-test) threshold can help to identify the best trade-off between cost and diagnostic accuracy.

Limitations of this study are described below. First, all patients in the heart disease dataset received three non-invasive cardiac tests and complete clinical and ECG data were collected for these patients. As not all patients in clinical practice would be expected to receive a comprehensive battery of non-invasive tests, the projected reduction in number of diagnostic tests ordered (and the associated costs) attributable to use of the ODPF algorithm may be optimistic. In fact, this limitation applies to most datasets in our analysis (except for the hepatitis dataset). Second, we did not evaluate the recommended sequences of diagnostic tests selected by the ODPF algorithm to determine whether these test sequences would be clinically acceptable to practicing physicians. To address this issue, one may set appropriate constraints (identified by clinician experts) to create viable test sequences in clinical practice. Third, the source dataset includes patients who were referred to a single center for diagnostic evaluation, and it is unclear how the ODPF algorithm would perform in an unselected, contemporaneous samples of patients with suspected CAD or thyroid disease.

In practice, a physician may need to obtain more tests to confirm diagnosis when the post-test probability marginally crosses the threshold. ODPF can also assist the decision of identifying the most effective or cost-effective test by using an appropriate LSFS function. We acknowledge that ODPF is designed to identify the minimum number of diagnostic tests needed to exceed a pre-determined probability threshold, but that other non-probabilistic considerations (e.g., the consequences of incorrect diagnosis, clinician regret) tend to determine whether additional testing is pursued in practice.

ODPF is a decision-support tool that shows disease probabilities and actively identifies the minimum set of tests most likely to confirm a diagnosis. ODPF needs input from users in several ways. Thresholds are the decision cut-points for ODPF and have to be determined based on the real-world situation. When no confident diagnosis can be made (i.e., no disease probability cross a threshold), physicians need to make the final diagnosis based on the final probability estimate. Human input can also be incorporated in the form of a manual override of the test choice, including rules that lead the test selection. For example, a certain test might be required under certain conditions.

Most datasets in this study have binary outcomes. Clinical practice, however, is often more complex. For example, a patient who presents with chest pain may have one of several possible diseases (e.g, myocardial infarction, pulmonary embolism, chest wall pain, etc.). Our future work aims to apply ODPF in more complex clinical settings, especially in the emergency department where the speed of diagnosis is of critical importance. Sometimes, a physician may order several tests simultaneously instead of one test at a time. Prediction of the contribution of genetic tests is a very interesting problem since these tests are expensive, and we can apply this algorithm to determine if one needs genetic tests and which tests. We will also evaluate different search methods to find the most promising group of diagnostic tests. One possibility is to use the minimum entropy function as an objective. This more general choice would reward tests that increase the “spread” of the posterior probabilities in general, as opposed to focusing on the most promising classes, as our uncertainty measures do.

In this project, we used a “0/1 function” to update instance weights; the training instances either match the query or they do not. We chose this function because we aimed to demonstrate the feasibility of using the lazy learning strategy to improve prediction. Alternatively, one could use a distance measure to compute instance weights, which would obviate the need to discretize continuous values. In future work, we plan to integrate a distance measure, possibly together with expert knowledge on the importance of individual tests, into the instance-weight mechanism. In addition, we plan to explore incorporating instance weights into the probability contribution function, or computing the posterior expectation given the known information of the query case in the expected uncertainty function.

The views expressed in this article are those of the author(s) and do not necessarily represent the views of the Department of Veterans Affairs.

References

[1] Elstein AS, Schwartz A. Clinical problem solving and diagnostic decision making: selective review of the cognitive literature. British Medical Journal 2002;324:729–32.
[2] Sox HC, Blatt MA, Higgins MC, Marton KI. Medical decision making. Stoneham, MA: Butterworth-Heinemann; 1988.
[3] Pauker SG, Kassirer JP. The threshold approach to clinical decision making. The New England Journal of Medicine 1980;302:1109–17.
[4] Moons KGM, van Es G-A, Deckers JW, Habbema JDF, Grobbee DE. Limitations of sensitivity, specificity, likelihood ratio, and Bayes' theorem in assessing diagnostic probabilities: a clinical example. Epidemiology 1997;80(1):12–7.
[5] Sackett DL, Haynes RB. Evidence base of clinical diagnosis: the architecture of diagnostic research. British Medical Journal 2002;324:539–41.
[6] Irwig L, Bossuyt P, Glasziou P, Gatsonis C, Lijmer J. Evidence base of clinical diagnosis: designing studies to ensure that estimates of test accuracy are transferable. British Medical Journal 2002;324:669–71.
[7] Whiting P, Rutjes AW, Reitsma JB, Glas AS, Bossuyt PM, Kleijnen J. Sources of variation and bias in studies of diagnostic accuracy: a systematic review. Annals of Internal Medicine 2004;1400(3):189–202.
[8] Shortliffe EH, Davis R, Axline SG, Buchanan BG, Green CC, Cohen SN. Computer-based consultations in clinical therapeutics: explanation and rule acquition capabilities of the MYCIN system. Computers and Biomedical Research 1975;80(4):303–20.
[9] Feigenbaum EA. The art of artificial intelligence. I. Themes and case studies of knowledge engineering. Technical report, Stanford, CA, USA; 1977.
[10] Watson ID, Basden A, Brandon PS. The client centered approach: expert system maintenance. Expert Systems 1992;90(4):189–96.
[11] Coenen F, Bench-Capon TJM. Maintenance and maintainability in regulation based systems. ICL Technical Journal 1992;76–84.
[12] Watson I, Marir F. Case-based reasoning: a review. The Knowledge Engineering Review 1994;90(4):355–81.
[13] Ishida Y. Using global properties for qualitative reasoning: a qualitative system theory. In: Sridharan NS, editor. Proceedings of international joint conference on artificial intelligence. San Mateo, CA: Morgan Kaufmann; 1989 . p. 1174–9.
[14] Menzies TJ. Knowledge elicitation: the state of the art. In: Chang SK, editor. Handbook of software engineering and knowledge engineering. River Edge, NJ: World Scientific Pub. Co.; 2002.
[15] Chi C-L, Street WN, Ward MM. Building a hospital referral expert system with a prediction and optimization-based decision support system algorithm. Journal of Biomedical Informatics 2008;410(2):371–86.
[16] Liau LC-K, Yang TC-K, Tsai M-T. Expert system of a crude oil distillation unit for process optimization using neural networks. Expert Systems with Applications 2004;260(2):247–55.
[17] Duda RO, Hart PE, Stork DG. Pattern classification, Second edition, New York: Wiley-Interscience; 2001.
[18] John GH, Kohavi R, Pfleger K. Irrelevant features and the subset selection problem. In: Cohen WW, Hirsh H, editors. Proceedings of the 11th international conference on machine learning. San Francisco, CA: Morgan Kaufmann; 1994. p. 121–9.
[19] Guyon I, Elisseeff A. An introduction to variable and feature selection. The Journal of Machine Learning Research 2003;3:1157–82.
[20] Bradley PS, Mangasarian OL, Street WN. Feature selection via mathematical programming. INFORMS Journal on Computing 1998;100(2):209–17.
[21] Aha D, Goldstone R. Concept learning and flexible weighting. In: Proceedings of the fourteenth annual conference of the cognitive science society. Hillsdale, NJ: Lawrence Erlbaum Associates; 1992. p. 534–9.
[22] Domingos P. Control-sensitive feature selection for lazy learners. Artificial Intelligence Review 1997;11:227–53.
[23] Howe N, Cardie C. Examining locally varying weights for nearest neighbor algorithms. In: Leake DB, Plaza E, editors. Proceedings of the second international conference on case-based reasoning research and development. London, UK: Springer-Verlag; 1997. p. 455–66.
[24] Park JH, Im KH, Shin C-K, Park SC. MBNR: case-based reasoning with local feature weighting by neural network. Applied Intelligence 2004;210(3):265–76.
[25] Saar-Tsechansky M, Melville P, Provost F. Active feature-value acquisition. Management Science 2009;550(4):664–84.
[26] Zheng Z, Padmanabhan B. On active learning for data acquisition. In: Zhong N, Yu PS, Wu X, Kumar V, Tsumoto S, editors. Proceedings of the 2002 IEEE international conference on data mining. Los Alamitos, CA: IEEE Computer Society; 2002. p. 562–9.
[27] Melville P, Saar-Tsechansky M, Provost F, Mooney R. Active feature-value acquisition for classifier induction. In: Bramer M, Wu X, Rastogi R, Morik K, editors. Proceedings of the fourth IEEE international conference on data mining. Los Alamitos, CA: IEEE Computer Society; 2004. p. 483–6.
[28] Kapoor A, Greiner R. Learning and classifying under hard budgets. In: Brazdil P, Jorge A, Gama J, Camacho R, Torgo L, editors. Proceedings of the 16th European conference on machine learning. New York: Springer; 2005 . p. 170–81.
[29] Domingos P. Metacost: a general method for making classifiers cost-sensitive. In: Madigan D, Chaudhuri S, Fayyad U, editors. Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining. New York, NY: Association for Computing Machinery; 1999 . p. 155–64.
[30] Turney PD. Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. Journal of Artificial Intelligence Research 1995;2:369–409.
[31] Ling CX, Yang Q, Wang J, Zhang S. Decision trees with minimal costs. In: Brodley CE, editor. Proceedings of the twenty-first international conference on machine learning, Banff, Canada. New York, NY: Association for Computing Machinery; 2004. p. 69.
[32] Gorry GA. Strategies for computer-aided diagnosis. Mathematical Biosciences 1968;2:293–318.
[33] Heckerman DE, Horvitz EJ, Nathwani BN. Toward normative expert systems. Part I. The Pathfinder project. Methods of Information in Medicine 1992;310(2):90–105.
[34] Heckerman DE, Nathwani BN. Toward normative expert systems. Part II. Probability-based representations for efficient knowledge acquisition and inference. Methods of Information in Medicine 1992;310(2):106–16.
[35] Guyon I, Weston J, Barnhill S, Vapnik V. Gene selection for cancer classification using support vector machines. Machine Learning 2002; 460(1–3):389–422.
[36] Aha DW. Lazy learning. Norwell, MA: Kluwer Academic Publishers; 1997.
[37] Atkeson CG, Moore AW, Schaal S. Locally weighted learning. Artificial Intelligence Review 1997;11:11–73.
[38] Vapnik V. The nature of statistical learning theory. NY: Springer; 1995.
[39] DeGroot MH, Fienberg SE. The comparison and evaluation of forecasters. The Statistician 1983;32:12–22.
[40] Zadrozny B, Elkan C. Transforming classifier scores into accurate multiclass probability estimates. In: Keim D, Hand D, Ng R, editors. Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. New York, NY: Association for Computing Machinery; 2002. p. 694–9.
[41] Niculescu-Mizil A, Caruana R. Predicting good probabilities with supervised learning. In: Raedt LD, Wrobel S, editors. Proceedings of the 22nd international conference on machine learning. New York, NY: Association for Computing Machinery; 2005. p. 625–32.
[42] Platt J. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Scholkopf B, Smola AJ, Bartlett PL, Schuurmans D, editors. Advances in large margin classifiers. Cambridge, MA: MIT Press; 1999. p. 61–74.
[43] Hsu C-W, Lin C-J. A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks 2002;130(2):415–25.
[44] Asuncion A, Newman DJ. UCI machine learning repository; 2007. Accessed at http://www.ics.uci.edu/∼mlearn/MLRepository.html.
[45] Detrano R, Janosi A, Steinbrunn W, Pfisterer M, Schmid J, Sandhu S, et al. International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology 1989;64: 304–10.
[46] Michie D, Spiegelhalter DJ, Taylor CC, editors. Machine learning, neural and statistical classification. Upper Saddle River, NJ: Ellis Horwood; 1994.
[47] Quinlan JR, Compton PJ, Horn KA, Lazurus L. Inductive knowledge acquisition: a case study. In: Quinlan JR, editor. Proceedings of the second Australian conference on applications of expert systems. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.; 1987. p. 137–56.
[48] Smith J, Everhart J, Dickson W, Knowler W, Johannes R. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Greenes RA, editor. Proceedings of the symposium on computer applications and medical care. New York: IEEE Computer Society; 1988. p. 261–5.
[49] Diaconis P, Efron B. Computer-intensive methods in statistics. Scientific American 1983;248:116–30.
[50] Fayyad UM, Irani KB. On the handling of continuous-valued attributes in decision tree generation. Machine Learning 1992;8:87–102.
[51] Predki B, Slowinski R, Stefanowski J, Susmaga R, Wilk S. Rose—software implementation of the rough set theory. Lecture Notes in Computer Science 1998;1424:605–8.
[52] Tan P-N, Steinbach M, Kumar V. Introduction to data mining. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.; 2005.
[53] Ling CX, Li C. Data mining for direct marketing: problems and solutions. In: Stolorz PE, Agrawal R, Piatetsky-Shapiro G, editors. Proceedings of the fourth ACM SIGKDD international conference on knowledge discovery and data mining. Menlo Park, CA: AAAI Press; 1998. p. 73–9.