MDP-based Cost Sensitive Classification using Decision Trees

Shlomi Maliah and Guy Shani

Software and Information Systems Engineering
Ben Gurion University, Israel
shlomima@post.bgu.ac.il, shanigu@bgu.ac.il

Abstract

In classification, an algorithm learns to classify a given instance based on a set of observed attribute values. In many real world cases testing the value of an attribute incurs a cost. Furthermore, there can also be a cost associated with the misclassification of an instance. Cost sensitive classification attempts to minimize the expected cost of classification, deciding after each observed attribute value, which attribute to measure next. In this paper we suggest Markov Decision Processes as a modeling tool for cost sensitive classification. We construct standard decision trees over all attribute subsets, and the leaves of these trees become the state space of our MDP. At each phase we decide on the next attribute to measure, balancing the cost of the measurement and the classification accuracy. We compare our approach to a set of previous approaches, showing our approach to work better for a range of misclassification costs.

1 Introduction

Classification is the task of identifying the class of an instance based on a set of observed attributes. For example, we may conduct a set of tests for a patient to properly identify her condition, applying an appropriate treatment for that condition. In many cases some tests, such as measuring temperature and blood pressure are easy to conduct and incur little cost, while other tests such as gene profiling are much more costly. On the other hand, costly tests are often more informative than blood pressure, and can be more useful for diagnosis. In such cases it may be reasonable to collect measurements in an incremental manner, deciding on the next test after observing the results of the previous test.

In addition, in many applications there is a cost associated with an error on the final classification decision. In the diagnosis example above, an error in classification may cause the doctors to apply a wrong treatment, which may be both costly in hospital resources, as well as damaging to the patient. There are hence two different costs that must be considered — a cost for measuring the value of a attribute, and the cost of misclassification.

Classical classification algorithms ignore these costs, focusing on measuring the value of attributes that provide better accuracy. In many cases better accuracy results implicitly in reduced misclassification cost. Cost sensitive classification (Elkan 2001; Turney 1995; Lomax and Vadera 2013), on the other hand, takes costs directly into consideration.

For example, several researchers suggested to modify the splitting criterion of standard decision tree algorithms, such as C4.5, to take into account costs (Tan 1993; Norton 1989; Núñez 1991). Turney (1995) used a genetic algorithm to construct a decision tree, taking into account not only immediate costs, but also future costs.

In this paper we suggest Markov Decision Process (MDP) (Bellman 1957; Puterman 2014) as a useful tool for modeling cost sensitive classification. MDPs allow a classifier to choose which attribute to measure next taking into consideration the expected future sum of costs. MDPs inherently take into consideration future actions, rather than considering only the immediate, myopic gain.

We construct a set of decision trees, over all attribute subsets. These decision trees are learned in the standard manner, ignoring all costs, where each node in the tree is associated with an attribute to measure, and each outgoing edge is associated with a constraint over the observed value for that attribute. Leaves are associated with a final classification decision. The state space of our MDP is the set of leaves of all trees. When classifying the MDP state conforms to a single leaf in one of the trees, based on the set of attributes values that were observed thus far. The available MDP actions are to select an additional attribute to test, transitioning to a leaf in other tree, or to classify based on the known attributes.

After choosing an attribute to measure based on the MDP policy, we transition to a new leaf in another tree given the observed attribute value. The cost of attribute measurement actions are the costs of the tests, and the costs for the final classification decision are the misclassification costs.

Even though we construct multiple decision trees, we do not propose an ensemble approach (Banfield et al. 2007). Our approach can be described as constructing a single cost sensitive decision tree from multiple accuracy driven decision trees.

We provide experiments over a set of benchmarks (Lomax and Vadera 2013) showing that our MDP approach scales well over these benchmarks. We provide a set of experiments when varying the cost of misclassification. When the misclassification cost is very high, one will query all available attributes that provide useful information before making a decision. When the misclassification cost is very low, it is often better to make a decision based on the priors without paying the cost of measuring any attribute. We show that for a range of symmetric and asymmetric misclassification costs between these two extremes, our MDP approach outperforms all other methods.

2 Background

We now briefly review relevant background — decision trees, cost-sensitive classification, and MDPs.

2.1 Decision Trees

In classification, given an instance that specifies values for a set of attributes, the classifier must predict a class for that instance. Decision trees are perhaps one of the most popular classification model in machine learning literature, due to their predictive power, as well as their readability. The nodes of the tree are labeled an attribute. The outgoing edges from a node labeled an attribute a are labeled the various possible values of a. The leaves are labeled a class name.

To classify a new instance, one starts at the root, and traverses the tree towards one of the leaves. At each node labeled an attribute we observe the value of that attribute in the current example, and traverse the edge corresponding to that value. Upon reaching a leaf we classify the instance to the class associated with that leaf.

Classical decision tree induction algorithms, such as ID3 and C4.5, construct a decision tree incrementally, given a population of example instances labeled known classes. The algorithms start with the entire population and select an attribute that would split the population into different groups, such that the split maximizes some criteria. Typical criteria for preferring one attribute over another can be the information gain or the gain ratio. These algorithms take a greedy approach, choosing the attribute that splits the population best in a single step, ignoring future splits.

2.2 Cost-sensitive Decision Tree Learning

Classical classification algorithms assume that when given a new instance to classify, the values of all attributes are already known. This is identical to the case where the values can be acquired online, while classifying, incurring no additional cost. In addition, classical algorithms assume that all classification errors have an identical penalty.

These two assumptions are often invalid in many real world problems, such as sensor activation in cellphones, car failure diagnosis, and medical diagnosis. For example, in medical classification problems (Turney 1995), the patient diagnosis typically involves a sequence of medical tests. As the tests are costly both in terms of resources (money and time), as well as in inconvenience or even damage to the patient, the goal is to reduce these costs prior to a successful classification. A typical diagnosis process performs tests sequentially, and after observing the results of a test, makes a decision to classify, or to conduct another test.

In some cases test costs are not independent, that is, the cost for testing two attributes is lower than the sum of costs of testing each independently. For example, in medical diagnosis, there is a cost associated with extracting a blood sample, but once a blood sample has been extracted, testing additional properties of the blood may not require another sample, and thus has a reduced cost.

It is also natural in such problems to define a cost for misclassification, such as diagnosing the wrong disease, or diagnosing a disease in a healthy person. These misclassifications may result in unneeded expensive treatments. Also, misclassifying a sick patient as healthy may result in serious damage to the patient. One can also assign a reward for the correct classification of a disease, corresponding to the improvement in the quality of life. Different misclassifications may incur different costs, and different correct classifications may incur different rewards.

Given these two types of costs, it is more reasonable to measure the expected classification cost, rather than the accuracy of the model. Classical classification algorithms that ignore these costs may result in relatively high expected costs. One can adapt arbitrary classification algorithms to be cost sensitive (Zadrozny, Langford, and Abe 2003; Domingos 1999). Alternatively, one can construct a cost-sensitive algorithm directly, as we do.

Early cost-sensitive decision tree induction algorithms, such as CS-ID3, IDX, and EG2 take a greedy approach, choosing an attribute given the myopic expected test cost and gain. ICET (Turney 1995) uses a genetic algorithm to learn a decision tree that minimizes the expected cost. Genetic algorithms, as any local search, tend to achieve a local minima, and may hence produce suboptimal classifiers. MetaCost (Domingos 1999) uses bagging to construct several decision trees over different samples of the data, relabeling examples to their minimal cost class, and then retraining.

Lomax and Vadera (Lomax and Vadera 2011) conduct a comprehensive comparison of cost-sensitive, showing that Turney's ICET is still among the best preforming algorithms, and that IDX (Norton 1989) also preforms well.

We take a maximizing approach, where positive rewards represent desirable outcomes, and negative rewards represent costs and undesirable outcomes, and the goal of the classifier is to maximize the expected reward, rather than minimize the expected cost of classification. More explicitly, we assume a negative reward (cost) $r_{a|P}$ for testing the value of an attribute a given the set of already tested attributes P, and a negative reward $r_{c,c'}$ for misclassifying an example of class c' as class c, and a positive reward $r_{c,c}$ for correctly classifying an example of class c.

2.3 Markov Decision Processes

A Markov decision process (MDP) (Bellman 1957; Puterman 2014) is a decision making algorithms, designed to take into account future decision while deciding on the next action. Formally an MDP is a tuple $\langle S, A, tr, R, H\rangle$ where S is a set of states, A is a set of actions, tr is a transition function, R is a reward function, and H is the planning horizon — the maximal amount of steps before the execution terminates.

The state space S models all the relevant features for making a decision. The system is always at exactly one state, and this state is known to the decision maker. The action set A models the actions that are available to the decision maker. At each phase the decision maker observes the current state, and decides on an action. The system then transitions to a new state, and the process is repeated.

The tr function models stochastic state transitions, that is, $tr(s, a, s') = pr(s'|s, a)$ is the probability of transitioning to state s' given that action a was executed at state s. We assume the Markovian property, where only the current state influences the transition, i.e., $pr(s_{t+1}|s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}) = pr(s_{t+1}|s_{t-1}, a_{t-1})$. The reward function $R(s, a)$ models desirable outcomes, as well as action costs. A positive reward is given for a successful outcome, such as a correct classification, while a negative reward is used to denote either undesirable outcomes (e.g. misclassification), or action costs (e.g. test costs). Given the Markovian property, rewards depend solely on the action and the state prior to the action, ignoring previous states and actions.

A solution to an MDP is typically expressed as a policy $\pi : S \to A$ — a mapping from states to actions. An optimal policy maximizes the expected sum of rewards: $\sum_{i=0..H} r_i$. The value iteration algorithm computes a value function $V : S \to R$, assigning a value to a state, modeling the expected sum of rewards from that state following an optimal policy.

One can compute a value function iteratively using the Bellman backup (Bellman 1957):

$$V_1(s) = \max_a R(s, a) \quad (1)$$

$$Q_{i+1}(s, a) = R(s, a) + \sum_{s' \in S} tr(s, a, s')V_i(s') \quad (2)$$

$$V_{i+1}(s) = \max_{a \in A} Q_{i+1}(s, a) \quad (3)$$

where V_i models the expected sum of rewards with i steps to go. Given cyclic transitions value iteration eventually converges, but for acyclic transitions, computing values in reversed topological order requires only a single iteration. Given a value function V we can define a policy:

$$\pi(s) = argmax_{a \in A} R(s, a) + \sum_{s' \in S} tr(s, a, s')V(s') \quad (4)$$

mapping each state to an action to execute.

MDPs have been previously suggested for the related task of dynamic feature selection (He, Daumé III, and Eisner 2012), where the goal is to collect a relatively small number of features to increase classification speed.

2.4 Reinforcement Learning

When the MDP parameters, the transitions and the rewards, are unknown, one cannot use value iteration directly. Computing a policy under unknown model parameters is typically known as reinforcement learning (RL)(Sutton and Barto 1998). There are two main approaches to RL; In model-based RL, the model parameters are learned from the data, allowing us to use value iteration. In model-free RL the value function is computed directly without learning the model parameters first. Perhaps the most popular model-free method is Q-learning (Watkins and Dayan 1992), where after executing an action a, at state s, receiving a reward r and transitioning to state s', the value function is updated:

$$Q(s, a) = (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \max_a Q(s', a)) \quad (5)$$

$Q(s, a)$ is typically called the Q function.

3 MDP-based Classification

We now describe our classification method, which uses an MDP to decide on the next attribute to test, or which class to classify an example. Our method uses a set of decision trees, each over a different subset of attributes. During classification, at each step there is a single active tree, corresponding to the set of already observed attributes, and a single active leaf in that tree, defining the current MDP state. The MDP can decide to transition to another tree and leaf testing the value of another attributes, or classify the current example.

Let $\mathcal{A} = \{a_1, ..., a_n\}$ be the set of attributes, and let $\mathcal{C} = \{c_1, c_2, ..., c_m\}$ be the set of classes of the current classification problem. We begin learning a decision tree for every subset of $\mathcal{A}$, including the empty set. The decision trees are constructed using the standard C4.5 algorithm (Quinlan 2014). When constructing the trees we maintain in each leaf l of every tree the set of examples E_l that were associated with this leaf during learning.

In cases where there are too many attributes to construct decision trees for all subsets, and some attributes are significantly less costly than others, we can reduce the construction costs assuming that the less costly attributes are always tested. Let $\mathcal{A}_1 \subset \mathcal{A}$ be the set of costly attributes, and $\mathcal{A}_2 \subset \mathcal{A}$ be the set of low cost attributes, such that $\mathcal{A}_1 \cup \mathcal{A}_2 = \mathcal{A}, \mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$. For each subset $P \subset \mathcal{A}_1$ we learn a decision tree for $P \cup \mathcal{A}_2$. That is, we learn a decision tree for each subset of the high cost attributes, together with the set of low cost attributes.

3.1 Constructing the MDP

Let T_P be the decision tree over an attribute subset $P \in \mathcal{P}(\mathcal{A})$ — the power set of $\mathcal{A}$. Let L_{T_P} be the set of leaves of a tree T_P. Each state of the MDP corresponds to a leaf in one of the trees. In addition there is a single terminal state s_t. Hence, the set of states of the MDP is defined :

$$S = s_t \bigcup_{P \in \mathcal{P}(\mathcal{A})} L_{T_P} \tag{6}$$

When classifying an example e, the execution always starts with the single leaf of the empty set tree $T_\emptyset$. Hence, the start state is the single leaf of $T_\emptyset$.

There are two types of actions in our MDP — testing the value of an unknown attribute, and classifying the example. Testing the value of an attribute a transitions the MDP from a leaf l in tree T_{P_1} to a leaf l' in the tree T_{P_2}, where $P_2 = P_1 \cup \{a\}$. We abuse notation denoting a both an attribute and the action that tests the value of that attribute. We can describe the transition as a two phase operation. In the first phase, we deterministically move from T_{P_1} to T_{P_2}. Then, we use the values of all tested attributes in P_2 as well as a to traverse TP_2 towards a single leaf.

We can potentially transition from a leaf l in T_{P_1} to a number of leaves in T_{P_2}, given different values for the tested attribute. However, once the value for that attribute is known, the transition is deterministic to a single leaf. We denote $\tau(l, a, e)$ the deterministic transition function between leaves, i.e. $l' = \tau(l, a, e)$ is the leaf that is reached given the value of a for an example e at leaf l. The cost (negative reward) for executing a at a leaf in a tree over a set of attributes P is $r_{a|P}$.

The second action type is a classification action. We further abuse notation denoting c both a class, and the action of classifying an example as class c. Following such a classification action c, we transition to the terminal state with probability 1, and the reward depends on the misclassification cost for c — if the current example was successfully classified into class c, then the reward is $r_{c,c}$. If an example of class c' was misclassified as c, then the cost is $r_{c,c'}$.

All costs are given as part of the problem description.

3.2 Solving the MDP

We take here a model-free approach — instead of learning the transition and reward functions from the given examples, we compute a value function and policy directly. As can be seen, the transitions in our model are acyclic, as we always transition to a tree over a set of attributes which is a superset of the already observed attributes, or to the terminal state. We can hence construct the value function, starting from states corresponding to the leaves of the tree built using the entire attribute set $\mathcal{A}$, moving backwards. This means that at iteration i we update only the states associated with the trees of attributes sets of size $n - i$. There is never a need to update the value of a state twice, making the computation of the value function in our case extremely efficient.

For the states corresponding to the leaves of the tree $T_\mathcal{A}$, the decision tree over the entire attributes set, where the only available actions are classification actions — we set the value function given the average misclassification cost. That is, for each leaf of $T_\mathcal{A}$ and each classification action c:

$$Q(l, c) = pr(c|l) r_{c,c} + \sum_{c'} pr(c'|l) r_{c',c} \tag{7}$$

where $pr(c|l)$ is the maximum likelihood estimator computed using the set of examples associated with the leaf l.

We now compute a value function for trees over smaller sets of attributes, starting with the leaves of the trees over $|\mathcal{A}| - 1$ attributes and moving down to the single leaf of the tree over the empty set $T_\emptyset$. For leaves of a tree over an attribute set P, such that $|P| = i$, we use the value function computed for leaves of trees over attribute sets $P' = P \cup \{a\}$ for $a \in \mathcal{A} \setminus P$, where $|P'| = i + 1$. We use the already computed value function of the larger trees to estimate the future rewards for each example in the currently evaluated leaf. Given an example e and a leaf l for which the value function was already computed, we can define the sum of future rewards for e as follows:

$$\begin{aligned} V(e, l) &= r_{c_e, c'} : c' = \pi(l) && (8) \\ V(e, l) &= r_{a|P} + V(e, \tau(l, a, e)) : a = \pi(l) && (9) \end{aligned}$$

where c_e is the class of the example e, P is the set of attributes in the tree of leaf l, and $\tau(l, a, e)$ is the new leaf in the tree over $P \cup \{a\}$ that is reached for example e after testing the value of a.

We compute the Q function and the optimal policy for a leaf l in the tree T_P, after computing the optimal policy for all leaves in the trees for each P' s.t. $|P'| > |P|$ using:

$$\begin{aligned} Q(l, c) &= pr(c|l) r_{c,c} + \sum_{c'} pr(c'|l) r_{c',c} && (10) \\ Q(l, a) &= r_{a|P} + \frac{\sum_{e \in E_l} V(e, \tau(l, a, e))}{|E_l|} && (11) \\ \pi(l) &= \operatorname*{argmax}_{act \in \mathcal{A} \cup C} Q(l, act) && (12) \end{aligned}$$

After the policy π was computed for a leaf l, we can compute $V(e, l)$ for all examples in that leaf.

If we cache the computed value for $V(e, l)$, the computation time for a tree T_P is $\sum_{l \in leaves(T_P)} O((|C| + (|\mathcal{A}| - |P|)) \cdot |E_l| = O((|C| + (|\mathcal{A}| - |P|)) \cdot |E|)$, where E is the set of all examples. An upper bound for the entire computation of the value function is $O(2^{|\mathcal{A}|} \cdot (|C| + |\mathcal{A}|) \cdot |E|)$.

We note that our transitions are not truly Markovian. The non-Markovian behavior manifests when we compute the Q values for a leaf l based on the policy of

Table 1: A set of example instances, their attribute values, and their class.

	a_1	a_2	a_3	C
e_1	8	7	4	c_2
e_2	9	7	4	c_2
e_3	12	8	4	c_2
e_4	4	6	4	c_1
e_5	0	5	4	c_1
e_6	-5	7	4	c_1
e_7	7	7	4	c_1
e_8	-7	7	2	c_1
e_9	5	7	4	c_2
e_{10}	7	7	4	c_2
e_{11}	7	0	2	c_1
e_{12}	7	8	2	c_1
e_{13}	-1	6	2	c_2
e_{14}	4	8	4	c_1
e_{15}	7	6	4	c_1
e_{16}	7	6	4	c_2
e_{17}	7	7	2	c_2
e_{18}	7	3	2	c_2

other leaves, whose distribution of examples is different than that of l. As such, our Markovian assumption is only a useful approximation, allowing us to construct a strong cost sensitive classifier.

3.3 Refining the MDP Online

After observing the value of some attributes, we can have a better estimation of the Q function. Specifically, for each leaf there exist a set of constraints over the values of some attributes in $\mathcal{A}$. After observing the values of some attributes, we can avoid considering possible future leaves whose constraints are violated the observed values of these attributes. We remove these leaves from the state space, and recompute $\pi(l)$, ignoring examples that fall into one of the removed leaves. As the computation of the Q function is very fast, this recomputation does not pose a significant barrier, and we repeat it after every observation of an attribute value. After classification, we reset the trees back to their original state, undoing all the leaf removals, so that future classifications use the original, rather than the pruned trees.

3.4 Example

We now demonstrate our methods over a given set of example instances, specified in Table 1. Figure 1 shows the set of decision trees that were constructed using these examples. In some cases, the trees are identical, because learning the value of a single attribute does not sufficiently help the prediction problem. For example, learning only the value of a_3 does not significantly improve the class prediction, and the tree is identical to $T_\emptyset$.

The set of leaves of these trees is the set of the MDP states. Hence, our MDP has 18 states. From a leaf in the tree for a_1 (Figure 1b), for example, we can only apply 4 actions — (1,2) classify, for the two possible classes (c_1, c_2), (3) test the value of a_2, transitioning to the tree over $\{a_1, a_2\}$ (Figure 1e), (4) test the value of a_3, transition to the tree over $\{a_1, a_3\}$ (Figure 1f).

Figure 2 provides an example of the transition from the leaves of the tree $T_{\{a_1\}}$, using the action a_3, revealing the value of the attribute a_3. The transition is hence to the leaves of the tree $T_{\{a_1,a_3\}}$. In this example $\tau(l_3, a_3, e_1) = l_{11}$, and $\tau(l_3, a_3, e_{11}) = l_{10}$.

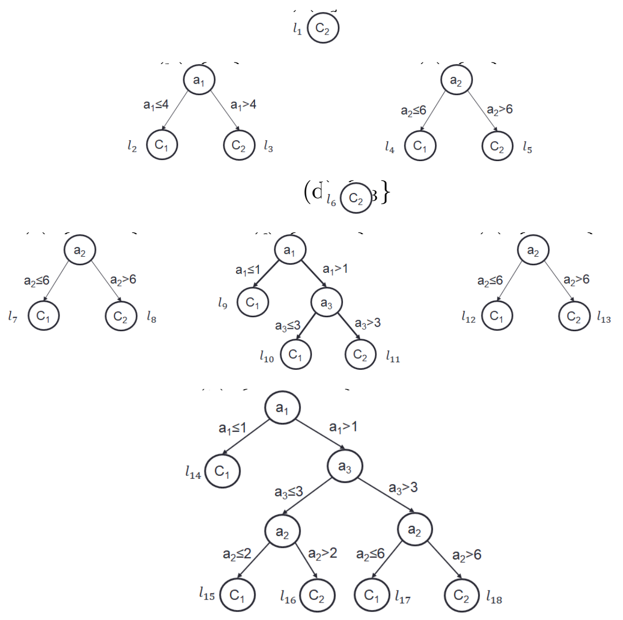


Figure 1: Constructed decision trees for all attribute subsets, constructed using the train data in Table 1.

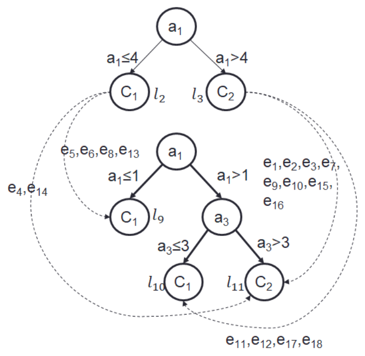


Figure 2: Possible transitions from the leaves in $T_{\{a_1\}}$ to the leaves in the tree for $T_{\{a_1,a_3\}}$, using the test a_3 action.

Let us further assume that the test cost for attribute a_i is i, and symmetric misclassification cost for all classes of 10 (and a reward of 10 for successful classification).

We can now begin computing the policy backward from the largest tree to the smaller ones. Looking at $T_{\{a_1,a_2,a_3\}}$, given symmetric misclassification costs it is always best to predict the most popular class in each leaf (specified in the leaves in Figure 1).

Turning our attention now to T_{a_1,a_3}, the only available non-classification action is testing a_2. For l_{11} we must compute the values using Equations 7 to 12, where $E_{l_{11}} = \{e_1, e_2, e_3, e_4, e_7, e_9, e_{10}, e_{14}, e_{15}, e_{16}\}$. We first compute the misclassification action value for l_{11}

using Equation 7:

$$pr(c_1|l_{11}) = \frac{4}{10} \quad (13)$$

$$pr(c_2|l_{11}) = \frac{6}{10} \quad (14)$$

$$Q(l_{11}, c_1) = \frac{4}{10} \cdot 10 + \frac{6}{10} \cdot (-10) = -2 \quad (15)$$

$$Q(l_{11}, c_2) = \frac{6}{10} \cdot 10 + \frac{4}{10} \cdot (-10) = 2 \quad (16)$$

We now compute $Q(l, a)$ using Equation 11. Examples e_4, e_{15}, e_{16} transition to l_{17}, where they are classified as c_1, which is correct for e_4, e_{15} but not for e_{16}. Examples $e_1, e_2, e_3, e_7, e_9, e_{10}, e_{14}$ transition to l_{18}, where 5 of them are correctly classified as c_2, and two (e_7, e_{14}) is misclassified. Computing $Q(l, a)$ we get:

$$Q(l_{11}, a_2) = -2 + \frac{5 \cdot 10 + 2 \cdot (-10) + 2 \cdot 10 + 1 \cdot (-10)}{10} = 2 \quad (17)$$

Hence, for l_{11}, $\text{argmax}_{act \in \mathcal{A} \cup C}\, Q(l_{11}, act) = c_2$ / a_2, so we are indifferent between classify as c_2 to test a_2 value.

For the leaf l_9, all examples transition to leaf l_{14}, with the same class distribution. Hence:

$$Q(l_9, c_1) = Q(l_{14}, c_1) = \frac{3}{4} \cdot 10 + \frac{1}{4} \cdot (-10) = 5 \quad (18)$$

$$Q(l_9, a_2) = -2 + 5 = 3 \quad (19)$$

and therefore $\pi(l_9) = c_1$.

For l_{10}, $E_{l_{10}} = \{e_{11}, e_{12}, e_{17}, e_{18}\}$, and the examples are evenly split between the classes, but after sensing the value of a_2, moving to l_{15} and l_{16}, we improve the classification accuracy, misclassifying only a single example (e_{12}):

$$Q(l_{10}, c_1) = Q(l_{10}, c_2) = \frac{2}{4} \cdot 10 + \frac{2}{4} \cdot (-10) = 0 \quad (20)$$

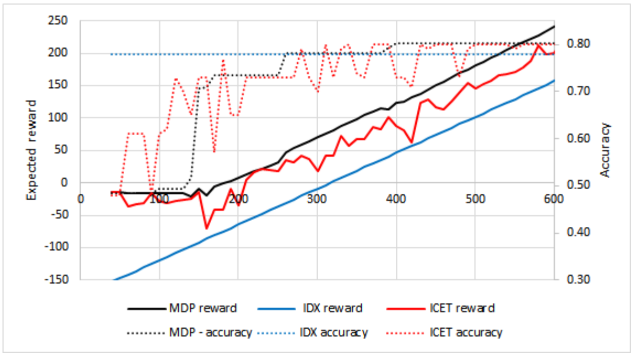
$$Q(l_{10}, a_2) = -2 + \frac{1 \cdot 10 + 2 \cdot 10 + 1 \cdot (-10)}{4} = 3 \quad (21)$$

and therefore $\pi(l_{10}) = a_2$.

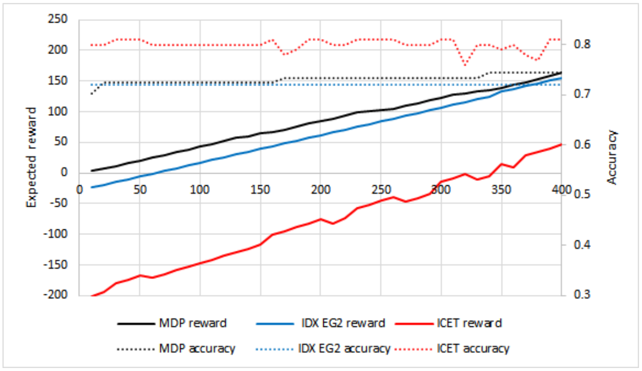
4 Empirical Evaluation

We now provide an experimental evaluation to compare our MDP-based cost sensitive classification method to other methods. We experiments with all domains from the recent cost sensitive survey (Lomax and Vadera 2013; 2011), comparing our approach to the leading methods in the survey.

Methods We base our implementation on Weka (Hall et al. 2009), except for ICET. The $C4.5$ algorithm that ignores costs provides an indication of the achievable accuracy, and an illustration on the expected cost when optimizing accuracy. We modified the Weka implementation of $C4.5$ to consider costs following 3 cost sensitive rules — CS-ID3, IDX, EG2. All three originally consider only test costs, and we modified them to consider misclassification cost when making a classification decision in a leaf. We also use Weka's implementation of MetaCost (Domingos 1999). In addition, we compare



(a) Glass



(b) Heart.

Figure 3: Comparing the expected reward and accuracy for varying misclassification costs (x-axis). In Heart, IDX and EG2 have identical performance and are reported together.

to ICET (Turney 1995), which showed the best performance in the survey (Lomax and Vadera 2011). ICET and MetaCost consider also the misclassification costs.

The experiments were run on a Windows 10 machine, $i5$ CPU, and 8GB RAM. Our method is implemented in $C\#$, ICET is implemented in C over a Linux virtual machine, and all other methods except for ICET are implemented in Java.

Procedure The ratio between the test costs and misclassification costs results in different behaviors. When the misclassification cost is much lower than the test costs, executing any tests results in suboptimal behavior. When the misclassification cost, at the other extreme, is much larger than test costs, it is often better to execute all informative tests. Hence, the behavior is most interesting in between.

We therefore experiment in each domain with 3 selected misclassification costs — one that is on the same order of the test costs, encouraging the algorithm to classify before executing many tests, one where the misclassification cost is sufficiently high to encourage the algorithm to execute many tests, and one where the misclassification cost is in between the two extremes. While we vary the misclassification costs we keep the test costs constant, as specified Lomax and Vadera (2011). MDP, ICET and MetaCost that consider misclassifica-

Table 2: Average classification reward (Rw) and average accuracy (Ac) with symmetric ($r_{c,c} = -r_{c,c'}$) misclassification costs. $r_{c,c'}$ shows the varying misclassification cost that we use, and $\overline{r_a}$ is the average cost for testing an attribute. MDP is our approach. IDX, CSID3, EG2 are the modification of $C4.5$ to cost sensitive splits. $C4.5$ is the original, cost insensitive algorithm.

			MDP		IDX		CSID3		EG2		C45		MetaCost		ICET	
Domain	$r_{c,c}$	$\overline{r_a}$	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw
	10	43.12327	0.68	3.52	0.96	-128.49	0.98	-136.11	0.96	-128.79	0.96	-137.49	0.96	-137.57	0.68	3.52
Car	100		0.69	35.23	0.96	-46.19	0.98	-50.14	0.96	-46.49	0.96	-54.82	0.96	-54.91	0.93	-47.00
	300		0.96	129.45	0.96	136.70	0.98	140.89	0.96	136.41	0.96	128.88	0.96	128.80	0.96	140.00
	100	41.71514	0.70	39.03	0.85	-94.21	0.84	-112.21	0.85	-94.21	0.85	-121.35	0.86	-173.10	0.79	-38.78
Breast	400		0.91	222.11	0.85	114.91	0.84	89.93	0.85	114.91	0.85	88.94	0.86	41.85	0.77	118.56
	600		0.92	390.56	0.85	254.33	0.84	224.68	0.85	254.33	0.85	229.14	0.86	185.15	0.82	230.01
	5	2.24	0.93	-0.07	0.99	-1.95	0.99	-6.65	0.99	-1.95	0.99	-9.64	0.99	-10.83	0.91	-1.00
Hepatitis	10		0.96	5.27	0.99	2.92	0.99	-1.79	0.99	2.92	0.99	-4.77	0.99	-5.96	0.92	1.30
	30		0.98	27.47	0.99	22.39	0.99	17.69	0.99	22.39	0.99	14.70	0.99	13.51	0.98	22.86
	10	23.24092	0.42	-1.50	0.97	-119.12	0.98	-109.81	0.98	-118.81	0.98	-109.40	0.98	-110.29	0.42	-1.50
Nursery	100		0.66	-11.14	0.97	-33.90	0.98	-22.84	0.98	-32.96	0.98	-22.35	0.98	-23.18	0.48	-18.40
	150		0.98	18.70	0.97	3.97	0.98	16.33	0.98	5.19	0.98	16.33	0.98	16.33	0.94	2.42
	5	1	0.64	1.36	0.87	-0.57	0.82	-0.89	0.87	-0.57	0.87	-0.57	0.86	-0.58	0.64	1.36
Tictactoe	10		0.84	4.70	0.87	3.08	0.82	2.29	0.87	3.08	0.87	3.08	0.86	3.04	0.83	3.40
	100		0.93	86.63	0.87	68.84	0.82	59.56	0.87	68.84	0.87	68.84	0.86	68.19	0.83	62.00
	10	53.5927	0.68	-38.68	0.95	-141.54	0.99	-163.93	0.98	-148.00	0.99	-183.30	0.99	-183.30	0.98	-189.43
Wine	100		0.91	13.41	0.95	-60.39	0.99	-74.91	0.98	-60.95	0.99	-95.26	0.99	-95.26	0.99	-130.40
	500		0.99	394.92	0.95	300.26	0.99	320.72	0.98	325.93	0.99	295.99	0.99	295.99	0.99	265.50

tion costs during construction were trained separately for each classification cost matrix.

We use a standard 80% − 20% train-test split. Results are paired, that is, all algorithms were run on the same split. Dataset statistics are reported in Table 5. For the larger datasets — Heart, Wine and Hepatitis, we learned decision trees only over subsets of the 10 most costly attributes together with the low cost attributes.

Due to the lack of space, we present here just a part of the results. Complete tables and all graphs can be found in the supplementary material, showing that our method performs well over all domain and over a wide cost range.

Results We first experiment with symmetric rewards and costs, that is, the cost of a misclassification is the negative of the reward for a successful classification. We also assume here that all misclassifications have the same cost. Symmetric misclassification costs are convenient for all methods (including ours) that ignore these costs when constructing the trees, as for symmetric costs, accuracy is completely correlated with the misclassification cost.

Table 2 shows the average reward per classification, and the prediction accuracy for each classification method, given varying misclassification costs. It is obvious that the MDP approach has a much better reward than any other method, often sacrificing accuracy to achieve better reward. For medium misclassification costs, which arguably require the most sophisticated balancing of accuracy and cost, our method is always best. It is also always best at the extremes, except for one domain (Car).

$C4.5$ is almost never best. It is obviously worst at low misclassification costs, when running multiple test is not worthwhile. It is also not good when the misclassification costs are high, and all attributes are tested, because it does not order the tests with respect to the cost. The cost-sensitive algorithms based on $C4.5$ work well when misclassification costs are high, but preform badly when costs are low, and are also far from optimal for the medium costs.

Considering accuracy, ICET and MDP have reduced accuracy when the misclassification cost is low, because it is not worthwhile to test more attributes in order to improve accuracy. It is interesting to see, though, that the MDP method often has better accuracy than even $C4.5$, when the misclassification costs are sufficiently high to warrant expensive tests for improving accuracy. This is most pronounced on tictactoe, but occurs in other domains as well. We attribute the high accuracy to the MDP ability to consider all future tests, while $C4.5$ is making greedy decisions at each split.

Figure 3 compares the behavior of our MDP approach, ICET, and C4.5 methods as the misclassification costs (and rewards) increase. C4.5 based methods (IDX and EG2) ignore these costs, and hence maintain fixed accuracy. ICET fluctuates for different costs, which is typical for local search methods that often get stuck at local maxima. In Glass ICET is sometimes only slightly below the MDP. In Heart ICET finds a tree with excellent accuracy, but very low expected reward. IDX offers very low expected cost in Glass, but converges towards our MDP approach in Heart, when the misclassification costs are high, because in these cases it is advantageous to invest in additional testing for improving accuracy. The MDP accuracy curve contains several thresholds where the MDP policy changes to

Table 3: Average reward (Rw) and average accuracy (Ac) asymmetric misclassification costs. $r_{c,c'}$ shows the range from which misclassification costs were randomly selected, and $\overline{r_a}$ is the average cost for attribute testing. IDX, CSID3, EG2 are the modification of $C4.5$ to cost sensitive splits. $C4.5$ is the original, cost insensitive algorithm.

			MDP		IDX		CSID3		EG2		C45		MetaCost		ICET	
Domain	$r_{c,c}$	$\overline{r_a}$	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw	Ac	Rw
Car	15-45	43.12327	0.68	22.61	0.95	-103.72	0.97	-110.49	0.95	-104.01	0.96	-112.60	0.96	-112.55	0.68	22.61
	25-155		0.23	6.57	0.96	-74.25	0.97	-79.73	0.96	-74.54	0.95	-81.62	0.95	-82.78	0.68	-7.84
	300-480		0.96	176.15	0.95	182.34	0.97	188.92	0.95	182.05	0.96	176.36	0.96	176.36	0.95	138.0
Breast	350-400	41.71514	0.75	92.24	0.74	12.01	0.72	32.11	0.74	12.01	0.74	44.03	0.74	45.29	0.73	37.14
	350-450		0.76	212.04	0.74	128.83	0.72	143.28	0.74	128.83	0.74	174.50	0.74	174.49	0.73	155.3
	500-800		0.66	148.53	0.74	42.19	0.72	65.99	0.74	42.19	0.74	81.21	0.74	82.46	0.75	94.10
Hepatitis	0-9	2.24	0.95	2.94	0.99	1.13	0.99	-3.58	0.99	1.13	0.99	-6.56	0.98	-6.51	0.94	1.50
	0-20		0.97	9.00	0.99	6.40	0.99	1.69	0.99	6.40	0.99	-1.29	0.99	-2.48	0.94	8.00
	25-70		0.99	26.00	0.99	19.11	0.84	10.84	0.99	19.11	0.96	11.64	0.98	7.40	0.99	23.00
Nursery	0-70	23.24092	0.98	62.06	0.97	44.36	0.98	58.58	0.98	46.95	0.98	58.85	0.98	46.95	0.94	44.27
	10-130		0.59	46.32	0.97	0.67	0.98	16.72	0.98	3.25	0.99	16.99	0.98	3.25	0.55	15.40
	10-170		0.66	80.69	0.97	40.52	0.98	55.62	0.98	42.87	0.98	56.01	0.98	42.87	0.53	53.00
Tictactoe	0-13	1	0.95	54.00	0.87	45.44	0.82	41.31	0.87	45.44	0.87	45.44	0.86	45.58	0.83	44.00
	13-25		0.82	6.28	0.87	3.81	0.81	3.14	0.87	3.81	0.87	3.81	0.90	4.10	0.64	5.73
	35-70		0.95	10.50	0.87	8.17	0.82	6.45	0.87	8.17	0.87	8.17	0.87	8.17	0.83	7.65
Wine	100-200	53.5927	0.94	83.70	0.95	9.44	0.99	2.30	0.98	14.29	0.99	-19.53	0.99	-19.53	0.98	-25.2
	50-300		0.95	47.56	0.95	-32.20	0.99	-39.17	0.98	-29.04	0.99	-60.73	0.99	-60.73	0.99	-94.1
	70-500		0.92	184.78	0.95	118.84	0.99	119.95	0.98	134.78	0.99	93.70	0.99	93.70	0.99	64.45

Table 4: Model construction time (sec.) for all methods. For our MDP approach we also report classification time.

Domain	MDP Classification	MDP Construction	IDX	CSID3	EG2	C45	MetaCost	ICET
Car	0.29	6.4	0.015	0.243	0.022	0.017	0.004	8
Breast	0.38	8.81	0.003	0.122	0.005	0.003	0.002	4
Hepati	0.06	45.86	0.015	0.247	0.019	0.018	0.004	6
Nursery	3.9	100.4	0.023	0.34	0.03	0.03	0.004	50
Tictac	2.65	95.73	0.016	0.265	0.015	0.015	0.003	8
Wine	0.01	5.63	0.003	0.125	0.006	0.005	0.003	4

Domain	Attributes	Train	Test
Iris	4	500	250
Car	6	700	300
Glass	8	500	200
Nursery	8	6500	2500
Tictactoe	9	700	300
Breast	9	300	100
Heart	12	200	100
Wine	13	400	200
Hepatitis	17	700	300

Table 5: Dataset statistics

test more attributes, resulting in a significant increase in accuracy.

We also experimented with asymmetric misclassification costs. We choose the costs and rewards uniformly from 3 ranges — small, medium, and large — all with respect to the test costs, as above. Table 3 shows these results for the asymmetric costs. Here, our MDP approach is almost always the best method. All other methods, while performing well in some cases, perform very poorly in other domains. For example, ICET does well on hepatitis, but does not perform well on breast. Overall, this table demonstrates, again, the superiority of our MDP approach over all other methods in most cases. MetaCost does much better on the asymmetric than the symmetric misclassification costs, and is sometimes better than ICET, but is best only rarely.

Table 4 shows the model construction time of the various methods. As can be expected, ICET takes much longer than the C4.5 methods, and our MDP approach takes the longest. In classification, all methods that use a single tree operate very rapidly, while our approach that recomputes the MDP policy after each test (Section 3.3) is significantly slower. Still, in many applications, such as medical diagnosis, spending a few minutes constructing the model, and a few seconds during classification is reasonable, given the considerable improvement in expected reward.

5 Conclusion

In this paper we have suggested an MDP-based approach to cost sensitive classification, considering both test costs and misclassification costs. The MDP approach provides a sound method for reasoning about future actions, instead of considering only the myopic

We construct standard decision trees over all attribute subsets, and the leaves of these trees become the state space of our MDP. We explain how transition probabilities can be estimated from the data, and define the costs and rewards of the MDP. We explain how our method enforces the Markovian property, and how the value iteration algorithm can be augmented to take our relaxation into consideration.

We analyze the sensitivity of our MDP method for with all methods and all domains from (Lomax and Vadera 2011), showing that our MDP approach is al

Acknowledgments: Supported ISF Grant 933/13, the

References

Banfield, R. E.; Hall, L. O.; Bowyer, K. W.; and Kegelmeyer, W. P. 2007. A comparison of decision tree ensemble creation techniques. IEEE transactions on pattern analysis and machine intelligence 29(1):173–180.

Bellman, R. 1957. Dynamic programming. Courier Corporation.

Domingos, P. 1999. Metacost: A general method for making classifiers cost-sensitive. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 155–164. ACM.

Elkan, C. 2001. The foundations of cost-sensitive learning. In International joint conference on artificial intelligence, volume 17, 973–978. Lawrence Erlbaum Associates Ltd.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: an update. ACM SIGKDD explorations newsletter 11(1):10–18.

He, H.; Daumé III, H.; and Eisner, J. 2012. Cost-sensitive dynamic feature selection. In ICML Inferring Workshop.

Lomax, S., and Vadera, S. 2011. An empirical comparison of cost-sensitive decision tree induction algorithms. Expert Systems 28(3):227–268.

Lomax, S., and Vadera, S. 2013. A survey of cost-sensitive decision tree induction algorithms. ACM Comput. Surv. 45(2):16:1–16:35.

Norton, S. W. 1989. Generating better decision trees. In IJCAI, volume 89, 800–805.

Núñez, M. 1991. The use of background knowledge in decision tree induction. Machine learning 6(3):231–250.

Puterman, M. L. 2014. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons.

Quinlan, J. R. 2014. C4.5: programs for machine learning. Elsevier.

Sutton, R. S., and Barto, A. G. 1998. Reinforcement learning: An introduction, volume 1. MIT press Cambridge.

Tan, M. 1993. Cost-sensitive learning of classification knowledge and its applications in robotics. Machine Learning 13(1):7–33.

Turney, P. D. 1995. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. Journal of artificial intelligence research 2:369–409.

Watkins, C. J., and Dayan, P. 1992. Q-learning. Machine learning 8(3-4):279–292.

Zadrozny, B.; Langford, J.; and Abe, N. 2003. Cost-sensitive learning by cost-proportionate example weighting. In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, 435–442. IEEE.