

Cost-Sensitive Fault Remediation for Autonomic Computing

Michael L. Littman and Thu Nguyen and Haym Hirsh

Department of Computer Science
Rutgers University, Piscataway, NJ

Eitan M. Fenson and Richard Howard

PnP Networks, Inc., Los Altos, CA

Abstract

We introduce a formal model of cost-sensitive fault remediation, derive an exact algorithm for solving the special case of deterministic observations, and demonstrate it on two example problems. This effort is part of two self-healing software projects that attempt to use collected data for better decision making in emerging autonomic systems.

1 Introduction

Motivated by our ongoing project on self-healing in autonomic systems, specifically diagnosis and repair, we introduce a formalism for *cost-sensitive fault remediation* (CSFR). In CSFR, a decision maker is responsible for repairing a system when it breaks down. To narrow down the source of the fault, the decision maker can perform a *test* at some cost, and to repair the fault it can carry out a *remedial action*. A remedial action incurs a cost and either restores the system to proper functioning or fails. In either case, the system informs the decision maker of the outcome. The decision maker seeks a minimum cost *policy* for remediating the fault.

CSFR bears many similarities to cost-sensitive classification (CSC, see Turney 1995, Greiner *et al.* 1996). Like standard classification, CSC is concerned with categorizing unknown instances. However, CSC can be viewed as a kind of sequential decision problem, in which the decision maker interacts with a system by requesting attribute values. The decision maker’s performance is the total of the individual action costs plus a charge for any misclassifications. Diagnosis problems include hidden information—the identity of the fault state—making them a kind of partially observable Markov decision process (POMDP, Kaelbling *et al.* 1998). POMDPs can be very notoriously difficult to solve, although there is reason to believe that CSCs result in relatively benign POMDP instances (Zubek and Dietterich 2002, Guo 2002).

The main difference between CSC and CSFR, and the main novelty of our work, is that a decision maker in a CSFR model can use feedback on the success or failure of a remediation action to attempt an alternate repair. In CSC, classification actions end episodes, whereas in CSFR, episodes continue until the fault is repaired. Although a relatively small change, we feel it adds significantly to the degree of autonomy supported by the model.

In contrast to general POMDPs, faults in a CSFR remain constant until repaired. This simplifies the construction of

policies compared to the general case and may enable efficient approximations. Therefore, the CSFR model lies at an intermediate point on the spectrum from the simple CSC model to the general POMDP model.

2 Formal Problem Definition

A cost-sensitive classification problem can be described formally by a set of classes C , a prior probability distribution $\text{Pr}(c)$ over the elements $c \in C$, a function representing misclassification costs $m(c, c')$ for incorrectly classifying an instance of class c' as a member of class c , a set of tests T , a cost function $j(t, c)$ giving the cost of executing test $t \in T$ in class $c \in C$, and a naive Bayes model $b(t, c)$ providing the conditional probability of observing the outcomes 0 or 1 for test $t \in T$ given that the true class is c . Many related formalisms are possible; we describe this one because it strikes a balance between power and simplicity. The decisions made by a cost-sensitive classifier map a partial assignment of the values 0 and 1 to the tests in T to an unobserved test in T or an announcement of a class $c \in C$.

The CSFR model adds a few wrinkles to the CSC model. Instead of being forced to classify instances, a CSFR learner can select from among a set of remedial actions R . The misclassification costs m are replaced by costs $m(r, c)$ for taking remedial action $r \in R$ when the fault class is $c \in C$. In addition, the goal function $G(r, c)$ returns 1 if executing remedial action $r \in R$ in class $c \in C$ results in repairing the fault and zero otherwise. We assume that for every $c \in C$ there is some $r \in R$ such that $G(r, c) = 1$ (every fault state can be fixed). In our preliminary work, we have assumed deterministic observations, $b(t, c) \in \{0, 1\}$ for all $t \in T$ and $c \in C$, which is a substantial oversimplification. Even with this restriction, however, stochastic observations can be simulated by increasing the number of fault states in the model.

3 Optimal planning in CSFR

To provide a first optimal planning algorithm for the CSFR model, we took advantage of the fact that test outcomes were deterministic in our example problems. This means that there is a single set of test outcomes associated with each fault state. This fact simplifies computations. Specifically, at the beginning of the remediation process, the probability of a fault c is simply its prior. As observations are made, some faults are

inconsistent with the new information, and their probability becomes zero (the probability of the remaining faults is normalized to one). So, during an episode, every fault probability is either zero or is proportional to its prior probability.

We say that a fault state c is *active* if it is consistent with all test results from the current episode. At the beginning of a remediation episode, all fault states are active. As the episode progresses and assuming that the decision maker only selects tests that have differing outcomes for the currently active fault states, each decision results in a shrinking subset of possible active default states. It follows that the number of decision situations that the decision maker will face is bounded by the size of the power set of C . Although this can be a very large number, it is finite and small enough to allow sets of fault states with as many as perhaps 20 observations to be handled in our preliminary experiments.

We can find an optimal strategy via a dynamic programming approach. Let S be the power set of C , which is the set of belief states of the system. For each $s \in S$, $r \in R$, and $t \in T$, define the expected value of a test t in a belief state s as the expected cost of the test plus the value of the resulting belief state:

$$Q(t, s) = \begin{cases} \Pr(s_0)/\Pr(s) j(t, s_0)V(s_0) + \\ \Pr(s_1)/\Pr(s) j(t, s_1)V(s_1), \\ \text{if } \Pr(s_0) > 0 \text{ and } \Pr(s_1) > 0; \\ \infty, \text{ otherwise.} \end{cases}$$

Define the expected value of a remedial action r analogously:

$$Q(r, s) = \begin{cases} \Pr(q_0)/\Pr(s) m(r, q_0)V(q_0) + \\ \Pr(q_1)/\Pr(s) m(r, q_1), \\ \text{if } \Pr(q_1) > 0; \\ \infty, \text{ otherwise.} \end{cases}$$

The difference between these two formulae is that there is no recursion in the case of a successful remedial action. The value of a belief state is the minimum over all available choices: $V(s) = \min_{a \in T \cup R} Q(a, s)$. Here, $s_i = \{c | c \in S \text{ and } b(t, c) = i\}$ is the belief state following a test outcome of i and $q_i = \{c | c \in S \text{ and } G(r, c) = i\}$ is the belief state following a remedial action outcome of i . The quantities $\Pr(s)$, $j(t, s)$, and $m(t, s)$ are the probability and cost functions extended to belief states in the natural probability-weighted way. Specifically, let $\Pr(s) = \sum_{c \in s} \Pr(c)$, $j(t, s) = \sum_{c \in s} \Pr(c)j(t, c)$, and $m(t, s) = \sum_{c \in s} \Pr(c)m(t, c)$. The conditions that $\Pr(s_0) > 0$ and $\Pr(s_1) > 0$ and that $\Pr(q_1) > 0$ in the recursion ensure that the equations bottom out; no quantity is defined in terms of itself. This restriction rules out only suboptimal policies, but guarantees that the algorithm can be implemented simply and with definite time bounds.

Once these equations are evaluated, the correct action to take given a belief state s is the action a that minimizes $Q(a, s)$. Such a policy will optimally trade off actions to gain information, actions to repair, and the information gained from a failed attempt at remediation, in a way that minimizes the total expected cost of remediation.

4 Remediation Examples

To explore the CSFR formalism, we have examined several simplified models of faults based on real problems: repairing

a disk on a server, and restoring connectivity in a local-area network.

Our disk-repair scenario includes seven fault states (Permanent disk failure, Partial disk failure, Transient disk error, Software bug, Cable disconnected, Cables crossed, SCSI controller failure), four tests (Disk online?, cost 1; Disk ok?, cost 600; All cables plugged in?, cost 60; Other disk online?, cost 1), and five remedial actions (Restart software, cost 30; Operator replug any loose cables, cost 70; Operator replug all cables, cost 420; Replace disk, cost 900; Replace SCSI controller, cost 900). Note that, here, the operator can be directed to assist with remedial actions.

Here is the optimal policy for the costs we assigned:

Disk online?

yes: Restart software

success: Done (Problem was Software bug or Transient disk error)

failure: Replace disk (Problem was Partial disk failure)

no: Other disk online?

yes: Operator replug any loose cables

success: Done (Problem was Cable disconnected)

failure: Operator replug all cables

success: Done (Problem was Cables crossed)

failure: Replace disk (Problem was Permanent disk failure)

no: Replace SCSI controller (Problem as SCSI controller failure)

The resulting policy begins with a relatively cheap test to see if the troubled disk is accessible. It withholds more expensive actions (like replacing a disk) until later in the troubleshooting process when they are found to be unavoidable. Note that the policy does not distinguish between a software bug or a transient disk error; the system is brought back online by restarting the software in either case. Note also that replugging loose cables is used as a remediation (fixes a disconnected cable), but also as a kind of test (if it fails, further diagnosis and repair is necessary). It is this last property that is missing from the cost-sensitive classification model, necessitating the extension we propose.

Our second, larger, example problem is based on maintaining connectivity in a TCP/IP local area network in the presence of faults. The tests are a combinations of checking internal parameters like local network settings and external conditions like whether a specific website can be accessed. Note that the costs of a failure and a success are often quite different because of the delays associated with the “time out” period necessary to declare failure. Costs are approximated based on estimated times for the measurement or remedial action.

In the interest of brevity, we list the series of actions chosen and associated observations made when the fault is that local DNS Settings are wrong:

Can I ping the gateway? Yes. Can I reach the central monitoring site? No. Can the central monitoring cite reach Yahoo!? Yes. Are my DNS setting ok? No. Repair DHCP DNS.

To a first approximation, the policy executes a series of tests until one fails, at which point the appropriate remediation action is taken. This structure follows from the high expense of test failures.

5 Ongoing Work

We are also applying reinforcement learning (Sutton and Barto 1998, Kaelbling *et al.* 1996) to the problem of remediation. An effective reinforcement-learning system could eliminate the need for detailed CSFR models to be built by hand. It could also extend the CSFR model to allow for “destructive” tests that change the state of the system while observing it. An additional challenge arises if only partial instances are encountered during training episodes, with the result that the information needed to make accurate decisions is incomplete.

In general, the data needed to create accurate models is not gathered from functioning systems because of the lack of methodology for analyzing and exploiting it. We want to reverse this trend and to install CSFR-based decision makers in working systems, improving data gathering and decision making and providing data for constructing more sophisticated autonomic systems down the line.

References

- [Greiner *et al.*, 1996] Russell Greiner, Adam J. Grove, and Dan Roth. Learning active classifiers. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*, pages 207–215, 1996.
- [Guo, 2002] AnYua Guo. Active classification with bounded resources. At <http://cs.umass.edu/anyuan/publications/SSS402AGuo.ps>, 2002.
- [Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
- [Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.
- [Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.
- [Turney, 1995] Peter D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.
- [Zubek and Dietterich, 2002] Valentina Bayer Zubek and Thomas G. Dietterich. Pruning improves heuristic search for cost-sensitive learning. In *Proceedings of the International Conference on Machine Learning*, pages 27–34, 2002.