Data Min Knowl Disc (2008) 17:317–342
DOI 10.1007/s10618-007-0086-6

Classification trees and decision-analytic feedforward control: a case study from the video game industry

Michael Brydon · Andrew Gemino

Received: 21 November 2006 / Accepted: 11 December 2007 / Published online: 31 January 2008
Springer Science+Business Media, LLC 2008

Abstract The objective of this paper is to use a challenging real-world problem to illustrate how a probabilistic predictive model can provide the foundation for decision-analytic feedforward control. Commercial data mining software and sales data from a market research firm are used to create a predictive model of market success in the video game industry. A procedure is then described for transforming the classification trees into a decision-analytic model that can be solved to produce a value-maximizing game development policy. The video game example shows how the compact predictive models created by data mining algorithms can help to make decision-analytic feedforward control feasible, even for large, complex problems. However, the example also highlights the bounds placed on the practicality of the approach due to combinatorial explosions in the number of contingencies that have to be modeled. We show, for example, how the "option value" of sequels creates complexity that is effectively impossible to address using conventional decision analysis tools.

Keywords Data mining · Probability estimation trees · Decision tree analysis · Video game development · Real options

1 Introduction

Decision making in the video game development industry is known to be difficult (Barry et al. 2006; Walfisz et al. 2006). Much of the difficulty arises due to the long lag between game development decisions and market outcomes. Development firms

Responsible editor: Gary Weiss.

M. Brydon (✉) · A. Gemino
Faculty of Business Administration, Simon Fraser University, Burnaby, BC, Canada V5A 1S6
e-mail: mjbrydon@sfu.ca

must make important and irreversible decisions early in a game's lifecycle but do not observe the market's response until the games are on store shelves (Charne 2006). In addition, the market performance of a video game is difficult to predict (Gaume 2006). Although increasingly sophisticated statistical and data mining techniques can be used to construct predictive models, residual variance due to technological innovation, changing consumer tastes, and a myriad of other exogenous factors make significant prediction error inevitable. Taken together, the challenges created by time lags and uncertainties are formidable barriers to standard prescriptive modes of decision making (e.g., Simon 1977). As is often the case for new product development projects, the lag between decisions and results makes adaptive, feedback-based decision making impractical (McCarthy et al. 2006). However, anticipatory feedforward control is equally impractical without an accurate predictive model of the system's response to different development choices. Not surprisingly, decision makers in industries characterized by high uncertainty often eschew formal analysis and rely instead on informal, intuition-based decision techniques (Rahul 2006; Khatri and Ng 2000).

This paper examines the potential of "decision-analytic feedforward control" to exploit the coarse-grained probabilistic predictive models created by data mining. Decision analysis is a set of techniques for assigning expected utility values to alternative courses of action in the face of uncertainty. The problem that arises in practice is that the effort involved in building a realistic decision-analytic model of a problem can be significant. A potential means of reducing the modeling burden is to induce the many complex conditional probability estimates required for decision analysis from historical data. As prior research in the field of utility-based data mining has shown, techniques for creating classification models can be adapted to the task of probability estimation (Provost and Domingos 2003; Zhang et al. 2006).

The primary objective of this paper is to present a methodology for transforming data mining output into a complete decision-analytic model. A secondary objective is to assess the feasibility of applying the methodology to a large, complex, real-world decision problem. We use data from the video game industry and a commercial data mining package to induce a classification tree for predicting games sales. We then describe the transformations required to convert the classification tree into a decision-analytic cost/benefit model of game development decisions. The paper does not attempt to break new ground in data mining, but rather seeks to complement existing research by showing how improved data mining techniques can fit within the larger contexts of utility maximization and strategic decision making. The establishment of a clear path from data to decisions is critical because, without seamless integration between the two, firms are unlikely to make the necessary investments in data and data mining technology (Davenport et al. 2001).

2 Feedforward control for video game development

The economic stakes for firms in the video game development business are high. The projected revenue of the video game industry by 2,010 is $46.5 B compared to just over $100 B for the more established film entertainment industry (Kolodny 2006). Like

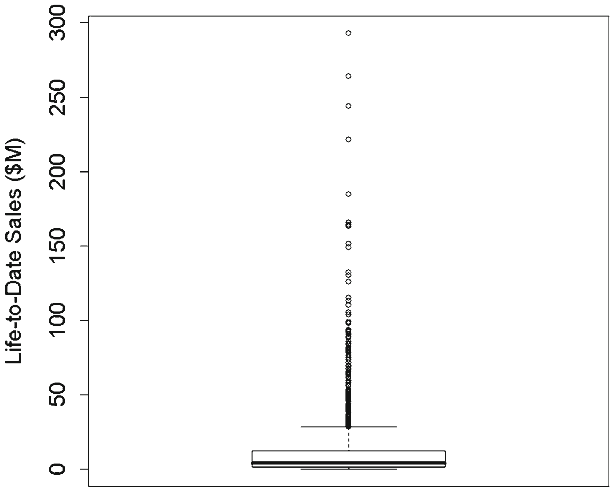

Fig. 1 A boxplot showing the distribution of LTD sale for 1,317 video games released between 2000 and 2004

the film industry, the video game industry is largely hit driven. Development costs for video games have risen dramatically as firms incorporate more complexity and detail in their quest for the next blockbuster. For example, Microsoft spent an estimated $40 M to develop *Halo 2*, which went on to earn more than $250 M in its first year (Reimer 2005). The game's sequel, *Halo 3*, was estimated to have cost only $30 M to develop (Pham and Friedman 2007) but grossed an estimated $170 M in its first day of sales.

Most games, however, do not do as well. A boxplot of the life-to-date (LTD) sales from our data set of 1,317 video games titles is shown in Fig. 1. The average revenue generated by a title across all three major consoles (Sony PlayStation2, Microsoft XBox, Nintendo GameCube) is $10.4 M; however, the median revenue for a game is only $3.9 M. Game studios thus face a significant risk that a new game will fail to recoup its development costs. This risk creates a strong incentive for firms to acquire the capability to predict which features of a game translate into sales and which features merely result in additional expense.

Many determinants of both a game's cost and revenues—such as its genre, target audience, licensing arrangements, and relationship to the developer's existing video game brands—are the result of decisions made early in the game development process. Other determinants of a game's performance, such as competition within a particular genre, are not completely within the developer's control. The task of setting the values for controllable and "semi-controllable" inputs in order to achieve a desired level of performance in the face of uncontrollable inputs is often expressed in the language of control theory. The control task facing decision makers in a video game development firm is to select game features so that the game achieves (or exceeds) a target level of profitability.

In a feedback control system—such as a thermostatically controlled heating system—the difference (or "error") between the desired output of the system (the "set

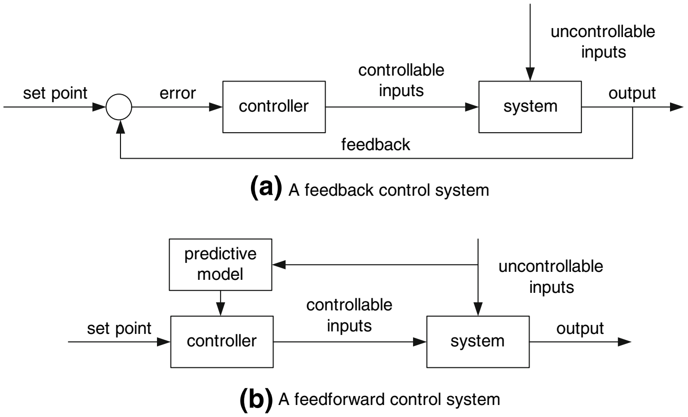

Fig. 2 Block diagrams for feedback and feedfoward control systems

point") and the actual output of the system is minimized by continuously monitoring the state of the system, measuring the error, and applying small compensatory control actions. The basic elements of a feedback control loop are shown in the block diagram in Fig. 2a. Such a control strategy is infeasible in environments such as video game development due to the long delay between control actions (development decisions) and information about the state of the system (the game's success in the market). Once the state of the system is known, it is too late to make adjustments to fundamental attributes of the game (Charne 2006).

An alternative to feedback control is feedforward control. The key difference between the two is that feedforward control is anticipatory rather than compensatory. As Fig. 2b shows, a feedforward control system monitors uncontrollable inputs, predicts deviations from the set point, and adjusts values of the controllable variables *before* an error condition arises. An accurate predictive model of the system's response to a range of controllable and uncontrollable inputs is critical to feedforward control because there is no mechanism for calibration with the actual system through feedback. But, as the relative scarcity of true feedforward control system in practice illustrates, creating accurate predictive models is difficult. The basic premise underlying the decision-analytic approach to feedforward control is that a coarse-grained and probabilistic predictive model is better for decision making than no model.

3 Decision trees and classification trees

In this paper, we use data mining output (in the form of a classification tree) as input into a decision-analytic feedforward controller (in the form of a decision tree). Confusingly, "decision tree" and "classification tree" are used interchangeably within the data mining and machine learning communities. However, the decision analysis community has long used the term "decision tree" to refer to a representational formalism

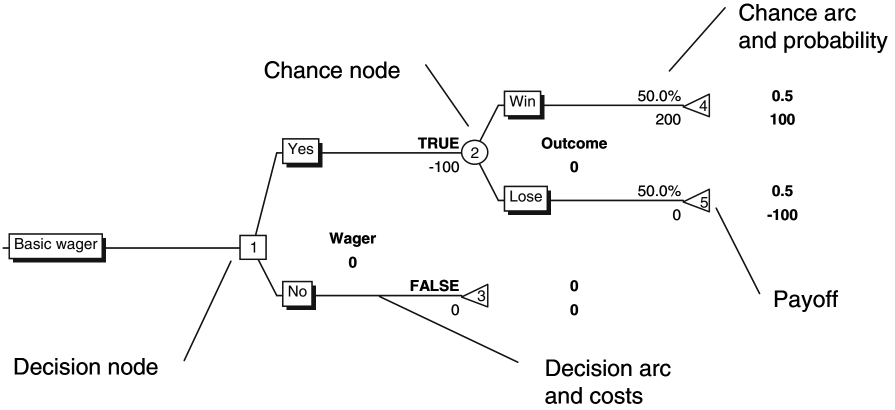

Fig. 3 Annotated output from commercial decision tree software. The tree describes a basic gamble consisting of an initial decision, a chance outcome, and payoffs

and related solution procedure that is distinct from a classification tree.[1] Given that the method for feedforward control described in this paper uses both decision and classification trees, it is useful to establish a clear distinction between the two.

3.1 Decision trees

A decision tree, as the term is used in decision theory, describes a connected directed acyclic graph (DAG) that is "attributed"—that is, its nodes and arcs are of distinct types with different semantics and operators (Schocken and Jones 1993). Specifically, a decision tree consists of three node types (chance, decision, and payoff) and two arc types (decision and chance). A decision tree is defined as a collection of nodes $i \in \{D \cup C \cup P\}$, where D, C, and P are decision, chance, and payoff nodes, respectively, and arcs $ij \in A_i$, which connect two nodes $i \neq j$. Each node in a decision tree has an expected value $v_i \in \Re$, which can be calculated by "rolling back" the complete tree (see below).

A simple decision tree for a basic gamble is shown in Fig. 3. The root of the tree is a decision node, $i \in D$, corresponding to a decision to wager on a particular team at a cost of \$100 or decline to wager, pay nothing, and win nothing. By convention, decision nodes are shown as squares. The arcs that emanate from a decision node represent mutually exclusive alternatives. Each alternative typically entails an immediate action cost, c_{ij}, which is assumed to be known. For example, the arc $\{1,2\}$, which corresponds to taking the bet in Fig. 3, involves a wager cost $c_{1,2} = -\$100$.

Node 2 in this gambling example corresponds to a chance node, which is shown by convention as a circle. The chance arcs $ij, i \in C$ represent the exhaustive and

[1] See decision analysis texts by Raiffa (1968) and Holloway (1979) for early uses of decision trees for decision making under uncertainty. A more recent text by Clemen (1996) discusses the relationship between decision trees and influence diagrams.

mutually exclusive outcomes of an uncertain event, such as a particular team winning or losing an important game. Each chance arc is assigned a probability estimate, p_{ij}, such that $\sum_j p_{ij} = 1$ for all i. Critically, these probabilities may be conditional on the information set at node i, which in turn depends on i's ancestor nodes.

A payoff node $i \in P$ is conventionally shown as a triangle and is used to terminate each path through the decision tree. The expected payoff value of each of these leaf nodes is known. For example, if the decision maker in Fig. 3 decides to wager and the wagered-upon team wins (Node 4), the payoff is $200. If the decision maker chooses not to wager, then the decision tree is asymmetric and terminates at Node 3 with a payoff of zero. The software used to create the decision tree in Fig. 3 shows the overall probabilities and net payoffs associated with each path to the right of the payoff nodes.

The distinction between different node and arc types permits decision trees to be evaluated using a dynamic programming procedure known as rollback. Rollback begins with the tree's payoff nodes and works backwards towards the root calculating the expected value of each node. The procedure used for the calculation depends on the node's type: In the most straightforward case, $i \in P$, the node's expected value is known. If $i \in C$, the node's value is calculated as the expected value of all successor nodes: $v_i = \sum_j p_{ij} v_j$ where $ij \in A_i$. Finally, if $i \in D$, the node's value is take to be the expected value (net of any action cost) of the best decision arc j, where $j = \arg\max_j (v_j - c_{ij})$ and $ij \in A_i$. In Fig. 3, the decision arcs selected during rollback are labeled "TRUE" whereas unselected arcs are labeled "FALSE".

A rolled-back decision tree thus provides both an optimal policy (mapping from decision nodes to actions) and an expected value for each decision arc. In cases in which the decision maker is not risk neutral, the monetary costs and payoffs in the decision tree can be replaced with their corresponding utilities (Raiffa 1968; Keeney and Raiffa 1976). Despite the long history of decision trees in decision analysis, there are two barriers to their widespread use in practice. The first and most important is the complexity of the trees. In the simplest case, in which each decision and chance node is binary ($|A_i| = 2,\ \forall i \in \{D \cup C\}$), and the tree is symmetrical, the number of leaf nodes is $2^{|D|+|C|}$. Path dependencies, such as conditional probabilities, preclude decomposition of the problem and thus an exponentially large decision tree may need to be built before rollback can commence. Pruning irrelevant branches to make an asymmetrical tree (as done along the "no bet" branch in Fig. 3) can reduce the effective size of the decision tree. However, pruning requires significant domain knowledge and effort on the part of the decision analyst (Raiffa 1968). The second barrier to the practical use of decision trees is the information requirement. In particular, estimating the conditional probabilities p_{ij} for all $i \in C$ can be extremely onerous. In some cases historical data is available and can be used for estimation. In other cases, decision analysts rely on the subjective estimates of experts.

3.2 Classification trees

A classification tree is also a connected DAG; however, classification trees are not normally attributed and do not serve the same purpose as decision trees. A classification tree is typically created by applying an inductive learning algorithm to a set of training examples, E. Each training example $\mathbf{x} \in E$ is a tuple $\mathbf{x} = \langle x_1 = v_1, \ldots x_n = v_n, r = l \rangle$, consisting of n feature value pairs plus a mapping for the single response variable r to a class label $l \in L$. For any new example, $\mathbf{x}' \notin E$, the classification tree provides a mapping $l \leftarrow f(\mathbf{x}')$.

A classification tree is constructed through a process of recursive partitioning of the training data (Breiman et al. 1984). Data mining algorithms use various splitting heuristics to estimate which variable x_i in E best "explains" the variation in the values of r. The training examples are then split into two subsets (assuming a binary classification tree) so that the homogeneity of each subset with respect to r is maximized. The path leading to each node thus defines a "rule" that consists of a conjunction of feature value pairs along the path. If we define the ancestors of i, $\mathbf{a}_i$, as all feature value pairs between the root node and node i inclusive, then the conditional probability distribution of r at i can be written: $p(r = l|\mathbf{a}_i)$.

Once the classification tree is constructed, the classifier need only find the path through the tree that is satisfied by the feature-value pairs in the unclassified example $\mathbf{x}'$. The class label of $\mathbf{x}'$ is determined by the distribution of training examples at leaf i. Specifically, $l = \underset{l \in L}{\arg\max}[p(r = l|\mathbf{a}_i)]$. To illustrate, consider the stylized classification tree for predicting video game sales shown in Fig. 4. Assume we have a new game for which all attributes values are known except that of the response variable, Sales. Sales of the new game can be estimated by selecting the path through the tree that is satisfied by the game's attribute values. Thus, a game with a value of Review Score = high would satisfy Node 3 (regardless of the values of the game's other attributes) and be classified as Sales = high based on the relative frequencies of high and low sales in the training data at that node.

Like decision trees, classification trees have the potential to grow exponentially large. The maximum number of leaf nodes in a classification tree based on n binary-valued attributes is 2^n. However, unlike decision trees, the worst-case size is seldom realized for classification trees. Most inductive learning algorithms include features

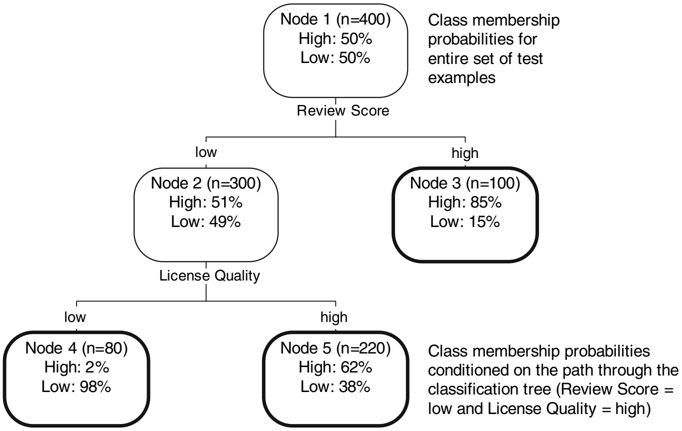
Node 1 (n=400)
High: 50%
Low: 50%
Class membership probabilities for entire set of test examples
Review Score
low
high
Node 2 (n=300)
High: 51%
Low: 49%
Node 3 (n=100)
High: 85%
Low: 15%
License Quality
low
high
Node 4 (n=80)
High: 2%
Low: 98%
Node 5 (n=220)
High: 62%
Low: 38%
Class membership probabilities conditioned on the path through the classification tree (Review Score = low and License Quality = high)

Fig. 4 A simple classification tree for predicting video game sales

such as significance thresholds for splits, minimum sample sizes for leaf nodes, and validation pruning that result in parsimonious classification trees. Recall in Fig. 4 that only a single feature was required to classify a new game if Review Score = high but two features were required if Review Score = low. The extent to which a final classification tree is smaller than the worst case is difficult to assess a priori. Tree size is determined to a large extent by characteristics of the training data such its signal-to-noise ratio and interdependencies between features.

3.3 The relationship between classification and decision making

Much of the prior research in data mining has focused on the classification problem. For example, in their survey of business applications of data mining, Apte et al. (2002) identify risk management, target marketing, and diabetes screening as classification tasks well-suited to data mining. Researchers have also focused on the cost of improving classification accuracy given non-zero data acquisition costs (e.g., Melville et al. 2005). The role of data mining in this paper, however, is not classification but rather feedforward control. As such, we are more interested in using the classification tree as the foundation for a control strategy rather than a means of accurately mapping class labels to new instances.

A naïve control strategy suggested by a classification tree might be to select the path through the tree that leads to the node with the most favorable probability distribution of outcomes. Returning to the classification tree in Fig. 4, a decision maker might simply set Review Score = high and thus maximize the probability of achieving high sales. However, such an approach is impractical for three reasons. First, the criteria used to select the most "favorable" probability distributions for the response variable needs to be defined. Possible criteria for selecting branches include highest expected value, highest $P(\text{Sales}=\text{high})$, lowest $P(\text{Sales}=\text{low})$, and so on. The choice depends on the risk tolerance of the decision maker. Second, some of the variables in the classification tree may not be fully controllable, thus limiting the decision maker's ability to simply choose a path. In this example, a development studio cannot fully control the reviews its games receive from independent critics. Third, classification trees for estimating benefits (e.g., revenues) typically do not contain information about the costs of achieving those benefits. Thus, the net expected benefit of using popular characters from a hit movie in a video game (License Quality = high in Node 5) cannot be determined without including the licensing costs payable to the owner of the characters.

A well-established method for reasoning about uncertain costs and benefits is to construct a decision tree. But, as noted above, an important shortcoming of decision tree analysis—and a reason why formal decision tree analysis is so rarely used for large decisions—is the cost of constructing the tree itself. Not only does the size of the tree increases exponentially with the number of chance and decision nodes, specifying the various costs, payoffs, and conditional probabilities requires detailed knowledge of the problem domain. Fortunately, data mining can address both these challenges. First, the same recursive partitioning techniques used to create classification trees can be used to construct probability estimation trees (PETs), which provide estimates of the probability of class membership.[2] Second, the classification tree provides a predictive model that is, in a sense, minimal. It identifies the features of the problem environment that (according to the data mining algorithm) are most relevant to the outcome. The decision tree can thus be limited to only these relevant variables.

We propose a two-phase strategy for feedforward control that exploits the ability of data mining algorithms to create concise predictive models. In the first phase, a large amount of historical data is used to create a probabilistic predictive model. In the second phase, the predictive model is used as the foundation for a decision tree. This clear demarcation between learning and decision making has several precedents in the artificial intelligence literature. For example, the data mining and decision analysis phases correspond to modeling and planning phases in the classic Sense, Model, Plan, Act (SMPA) architecture (c.f. Brooks 1991). The integration of learning and decision making is similar in spirit to reinforcement learning. The objective of a reinforcement learning (RL) algorithm is to maximize the expected value of its chosen actions given uncertainty about payoffs and causal relationships (Russell and Norvig 1995). In many RL contexts, such as robot control, the learner explores its unknown environment and makes incremental adjustments in response to what it has discovered. In the complex decision context addressed here, however, the lag between decisions and outcomes precludes conventional online RL techniques. Our approach is therefore closer to "batch" RL (Pednault et al. 2002). The primary difference between batch RL and decision-analytic feedforward control is that the latter uses a decision tree rather than a state-based Markov Decision Process (MPD) representation of the decision problem. The advantage of decision trees, as discussed in the following section, is that they are easily understood by managers and help to highlight the information gaps that typically emerge in a data mined predictive model.

4 Transformation of classification trees into decision trees

The procedure for transforming each node in a classification tree into a node in a decision tree depends on whether the node is a leaf or interior node. Recall that the data mining algorithm partitions each interior node i of the classification tree on a splitting variable $x_i = v_i$. Thus, the transformation of interior nodes further depends on whether the splitting variable is uncontrollable (*UC*), controllable (*CO*) or semi-controllable (*SC*). A classification tree node $i \in UC$ if the decision maker has no influence over the value of x_i. Uncontrollable variables typically correspond to states of nature (e.g., the weather) or the behavior of competitors in large markets in which strategic interactions are negligible. At the other extreme, $i \in CO$ if x_i is controllable. A variable is controllable if its value is fully determined by the actions of the decision maker. An example of a controllable variable in the video game context is NumPlatforms: the studio determines its strategy for porting games and there is virtually no uncertainty about the value of NumPlatforms once a porting decision has been made. A node in the classification tree is semi-controllable, $i \in SC$, if the value of x_i is

[2] Although this research uses data mining for probability estimation, not classification, we use the "classification tree" terminology throughout to avoid confusion.

partially determined by the actions of the decision maker. Semi-controllable nodes are extremely common in practice since almost all real-world decisions involve some amount of uncertainty regarding outcomes (ruling out $i \in CO$) and many seemingly uncontrollable events, such as the behavior of competitors, can be influenced through signaling and other actions (ruling out $i \in UC$). Finally, the leaf nodes of the classification tree must be dealt with differently since they are not split on an explanatory variable. Instead, each leaf node, $i \in LF$, provides a distribution of training examples with respect to the value of the response variable r.

The procedure for transforming each of these four types of classification tree nodes into corresponding decision tree representations is described in the following sections. Schematic representations of the transformation are shown in Fig. 5.

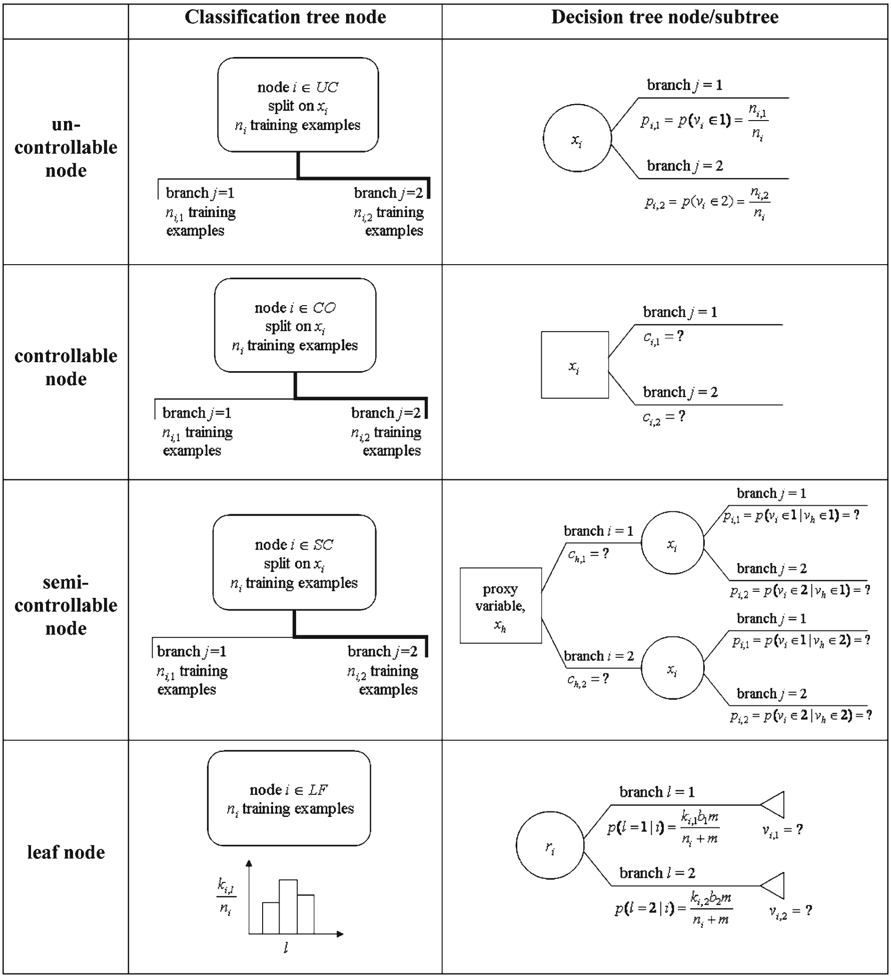
Classification tree node
Decision tree node/subtree
un-controllable node
node $i \in UC$ split on x_i n_i training examples
branch j=1 $n_{i,1}$ training examples
branch j=2 $n_{i,2}$ training examples
x_i
branch $j = 1$
$p_{i,1} = p(v_i \in 1) = \frac{n_{i,1}}{n_i}$
branch $j = 2$
$p_{i,2} = p(v_i \in 2) = \frac{n_{i,2}}{n_i}$
controllable node
node $i \in CO$ split on x_i n_i training examples
branch j=1 $n_{i,1}$ training examples
branch j=2 $n_{i,2}$ training examples
x_i
branch $j = 1$
$c_{i,1} = ?$
branch $j = 2$
$c_{i,2} = ?$
semi-controllable node
node $i \in SC$ split on x_i n_i training examples
branch j=1 $n_{i,1}$ training examples
branch j=2 $n_{i,2}$ training examples
proxy variable, x_h
branch $i = 1$
$c_{h,1} = ?$
branch $i = 2$
$c_{h,2} = ?$
x_i
branch $j = 1$
$p_{i,1} = p(v_i \in 1 | v_h \in 1) = ?$
branch $j = 2$
$p_{i,2} = p(v_i \in 2 | v_h \in 1) = ?$
x_i
branch $j = 1$
$p_{i,1} = p(v_i \in 1 | v_h \in 2) = ?$
branch $j = 2$
$p_{i,2} = p(v_i \in 2 | v_h \in 2) = ?$
leaf node
node $i \in LF$ n_i training examples
$\frac{k_{i,l}}{n_i}$
l
r_i
branch $l = 1$
$p(l = 1 | i) = \frac{k_{i,1} b_1 m}{n_i + m}$
$v_{i,1} = ?$
branch $l = 2$
$p(l = 2 | i) = \frac{k_{i,2} b_2 m}{n_i + m}$
$v_{i,2} = ?$

Fig. 5 Tranformations of classification tree nodes to decision tree nodes

4.1 Uncontrollable nodes

Consider a classification tree node $i \in UC$ that contains n_i training examples. The classification tree algorithm splits the examples into $j = 1, \ldots, J$ disjoint sets based on some threshold values of the splitting variable x_i. The number of training examples in each set (denoted n_{ij}) provides an estimate of the conditional probability of x_i taking on a value that falls within the bounds of the jth set. An uncontrollable node in a classification tree can thus be transformed directly into a chance node in a decision tree, $i \in C$, by creating a chance arc ij for each of the J sets (or branches) and setting the probability of each arc to $p_{ij} = \frac{n_{ij}}{n_i}$.

4.2 Controllable nodes

The transformation of a controllable node in the classification tree, $i \in CO$, to a node in the decision tree is straightforward: a new decision tree node $i \in D$ is created and a decision arc is added for each set $j = 1, \ldots, J$ of x_i. Unlike an uncontrollable node, the relative frequencies of the training examples for each child branch of $i \in CO$ are not used in the transformation since the decision tree's rollback procedure makes a deterministic commitment to one of the decision arcs. However, in order to select the best decision arc, the action cost, c_{ij}, along each arc must be known. The problem is that this cost information is not part of the classification tree.

An analogous issue of "missing" costs often arises in the context of cost-sensitive learning. In some cases the missing costs can be estimated from a superset of the training data. For example, Zadrozny and Elkan (2001) describe a problem in which the main learning objective was to estimate the probability that an individual with feature values $\mathbf{x}'$ would make a donation, $p(r = l|\mathbf{x}'), l \in \{donate, \neg donate\}$. However, to correctly estimate the cost of misclassification, the dollar amount of a donation, if any, had to be estimated. Linear regression was used to estimate donation amount $y(\mathbf{x}')$ using the same training data used for estimating donation probability.

In general, information about decision costs has to be acquired from other sources, such as historical cost data or expert opinion. In some cases the cost may be contingent on other variables, requiring that additional decision or chance nodes be added to the decision tree. As illustrated in Sect. 5, the primary advantage of decision trees as a representational formalism is that they permit information from diverse sources to be combined using the same set of constructs. The only requirement is that the newly acquired information be coarse-grained, discrete, exhaustive, and expressed in the language of decisions, chance, and payoffs.

4.3 Semi-controllable nodes

The transformation of node $i \in SC$ is more complex than the other node types because the probability that the variable x_i belongs to a branch j is conditional on the values of decision variables that are not represented in the classification tree. The transformation of a semi-controllable node therefore requires the addition of a new subtree that contains, at a minimum, a decision node based on a *proxy* variable, x_h, (Keeney and Raiffa 1976) and a chance node for each decision arc of the proxy decision node. The subtree is considerably more complex if the probability distribution of x_i is conditional on multiple proxy decision or chance variables.

The conditional probability of the various branches resulting from the split on x_i cannot be calculated from information contained in the classification tree (as they were for uncontrollable nodes) due to the addition of the proxy variable(s). As a consequence, the probabilities must be estimated from other sources of information. In addition, the proxy decision node(s) may introduce new action costs that must also be estimated. This procedure is more fully explained using an example in Sect. 5.2.

4.4 Leaf nodes

A leaf node in a classification tree is transformed into a chance node plus a payoff node to terminate each of the chance arcs. A real-valued payoff is assigned to each payoff node in order to seed the rollback of the decision tree. To illustrate the transformation, consider a leaf node, $i \in LF$ with n_i training examples. The number of examples at the node with label $r = l$ is k_{il} so the relative frequency of each class is $\frac{k_{il}}{n_i}$. It is important to note, however, that the learning algorithm's preference for homogeneous leaf nodes means that the relative frequencies are biased estimators of the true conditional probability of a particular outcome (Zadrozny and Elkan 2001; Provost and Domingos 2003). This bias can be reduced in several ways, including various smoothing transformations that shift the probability estimates towards less extreme priors. For example, Zadrozny and Elkan (2001) estimate the probability of each outcome in the leaf node as $p(l|i) = \frac{k_l bm}{n+m}$, where b is the base rate probability of l in the complete training sample and m is a tunable parameter. These smoothed probabilities are then used along the chance arcs of the decision tree.

5 Illustration: video game development

This section has two objectives. The first is to illustrate the transformations in Fig. 5 with specific examples that highlight some of the practical issues encountered during the transformation. The second is to demonstrate the feasibility of the approach by applying it to a complex, real-world decision problem. The primary risk of the proposed approach for probabilistic feedforward control is that the decision tree that results from transformation and augmentation of the classification tree is unmanageably large. The a priori risk is significant in this case because the potential number of chance and decision nodes required to adequately represent the problem in a decision tree is high. The data set used to train the classification tree has 29 explanatory variables, most of which have three categorical values. If all the variables are relevant to the final outcome and appear in the decision tree, the tree will have roughly 3^{29} leaf nodes. The advantage of the classification tree algorithm is that it discovers a subset of variables to serve as the foundation for a much smaller decision tree.

Given that we address only a single problem, we make no claim that the proposed approach is feasible in all cases. As in most data mining, the complexity of the final classification tree is problem-specific. Nor do we claim that the policies that result from our analysis are optimal for some or most video game developers. As we discuss below, the data available within the video game industry has important shortcomings that preclude an accurate predictive model. Instead, the video game problem is used here as a representative instance of the class of large strategic decision problems. The level of complexity we encounter—such as the number of explanatory variables and the degree of interdependence between variables—is likely similar to what might be encountered in other domains. We therefore argue that the video game problem provides an appropriate proof-of-concept problem.

5.1 The data set

The data used to train and test the classification tree was purchased by our industry partner, GameCorp,[3] from a market research firm that monitors sales in the video game industry. There are two advantages of using industry-level data to create a predictive model rather than the sales data from a single game developer or publisher. First, even the largest game studios release only a handful of games each year and a firm-specific data set would be too small to support inductive learning. Second, each studio tends to specialize in certain types of games, such as sport simulations or first-person shooters. Industry-level data provides much greater variation in sales as well as game attributes such as genre, target consoles, Entertainment Software Rating Board (ESRB) ratings, licensing arrangements, and game quality. The variation in the data makes it easier for the data mining software to identify relationships between the explanatory and response variables.

The core data set consists of 2,245 games developed for the PlayStation 2 (PS2), XBox, and GameCube consoles between 2000 and 2004. Approximately 20 features were included in the data set for each game, including genre, sub-genre, ESRB rating, console, brand, license (if any), publisher, release date, and the average review scores from a panel of game critics. We added a number of derived measures, including the prior sales of games within the same brand family and the number of concurrent releases of games within the same genre. A small number of candidate response variables were available in the data set, including life-to-date (LTD) Sales and LTD Units. Although life-to-date measures are ostensibly biased in favor of older games (games released long before the data set's cutoff date), the lifespan of most games is just a few months and only the most successful games remain on shelves for a year (Gaume 2006). To confirm, we regressed LTDSales on FirstYearSales, which is not biased in favor of older games, and discovered a high correlation ($R = 0.96$). We concluded that LTDSales was not overly biased in favor of older games. There was also the possibility of bias in favor of newer games due to price inflation since all price amounts were in nominal dollars. However, no adjustments were made to the sales data to account for inflation due to the relatively short time period under consideration and the lack of an accepted quality-adjusted price deflator for video games.

[3] The name of the company has been disguised.

5.2 Data transformations and additions

Several modifications of the data were required to make it more suitable for its ultimate role in a feedforward control strategy and to help manage growth of the decision tree. These modifications are summarized in the subsections below.

5.2.1 Incorrect granularity

The data set recorded features at the SKU (stock-keeping unit) level. As a result, games such as *ESPN Winter Sports 2002* were represented in the data three times: once each for the PS2, XBox, and GameCube versions. Although the different versions of a game differ slightly in terms of features, pricing, and even review scores, we combined all versions into a single game and aggregated features such as sales (sum across all platforms) and review score (maximum across all platforms). The NumPlatform = $\{1, 2, 3\}$ variable was added to the data set as were dummy variables for each target console.

5.2.2 High skewness

As the boxplot in Fig. 1 shows, the distribution of sales for the games in the data set is highly skewed. Since such skewness violates the assumption of normality made by linear regression techniques, it is standard practice in regression analysis to transform skewed variables to increase the symmetry of their distributions. Although no assumption of normality is made by information theoretic and χ^2-based splitting heuristics employed by data mining algorithms, we performed logarithmic transformations on all skewed variables (for example, all the sales-related variables) to provide the algorithm with more symmetric distributions (Pyle 1999).

5.2.3 Continuous variables

Although many data mining packages permit real-valued response variable (resulting in a regression tree rather than a classification tree), a discrete response variable is often easier for managers to interpret than a mean and standard deviation. Thus, all continuous variables were converted to categorical types with a domain size of at most three. In some cases GameCorp provided what they considered to be meaningful cut-off thresholds for the discretization. For example, a threshold of 85 points and above was defined as ReviewScore = high, between 70 and 85 points was defined as med, and scores lower than 70 points were deemed low. If no clear prior thresholds were provided, the continuous variables were divided into three equal quantiles. For example, the transformed response variable, LnLTDSales was partitioned into three categories $L = \{\text{low, med, high}\}$ such that the data set was balanced (a roughly equal number of training examples was in each category).

5.2.4 Nominal variables with large domains

The data set contained several nominal (categorical) variables with large domains. For example, License contained 109 different licensed brands used by video game companies, including *Star Wars, NASCAR*, and *Dora the Explorer*. The data mining package used in this research combines the values of nominal variables into clusters based on their ability to predict the response variables; however, such clusters rarely have any discernable business meaning. Moreover, the ultimate decision making value of predictive rules such as “IF License = NFL Football OR Star Wars THEN Sales = high” is limited since these licenses have already been granted. Our solution was to replace all large-domain nominal variables in the data set with variables that were more appropriate for decision making. For example, the License variable was replaced by a variable designed to capture the underlying *attractiveness* of the licensed brand. To measure attractiveness, we conducted our own market research. We presented a list of the 109 licensed brands in the data set to a sample of potential customers and asked them to rate the attractiveness of each license (irrespective of any video game connection) on a five-point Likert scale. A value of “N/A” was offered in case the participants were unfamiliar with the brand. The average rating on the five-point scale for each licensed game was added to the data set as the game’s LicenseScore value.

The Brand variable was also replaced by a derived variable designed to measure the brand’s underlying strength. Clearly, there is some overlap between the Brand and License variables in the data set. Brands licensed from outside the video game industry appear as both brands and licenses. However, not all brands are licenses. *Donkey Kong*, for example, is a strong brand that was developed within the video game industry and does not appear as a license. As the abundance of sequels in the video game industry shows, developers believe that the success of a game begets the success of follow-on games that use the same brand, themes, characters, and so on. We included two variables to capture the interdependency between games of the same brand. One variable, PriorSaleBrand, records the sales performance of the game’s immediate prequel (if any). The other variable, PriorSalesBrand, records the cumulative sales of *all* previous games (if any) of the same brand.

5.3 Data mining

We created our predictive model using SAS Enterprise Miner Version 9.1. Like most commercial data mining software, Enterprise Miner offers a choice of well-established splitting heuristics to recursively partition the set of training data into either a regression or classification tree (Haughton et al. 2003). We made only minor changes to the default data mining parameters provided by the program. For example, the maximum number of branches at each node was increased from two (binary splits only) to three in order to simplify partitions on the many three-valued (high, med, low) categorical variables in the data set. We also turned off validation pruning, in accordance with the PET recommendations in Provost and Domingos (2003).

The resulting classification tree for predicting video game sales is shown in Fig. 6. The relative frequencies of the response values in the nodes were provided by an independent hold-back sample of roughly 40% of the data. As the distribution of outcomes at the leaf nodes show, the information gain provided along each branch of the tree varies dramatically. At some leaf nodes, the frequency of the three sales categories (Quantile(LnLTDSales) = {high, med, low}) is roughly uniform, indicating

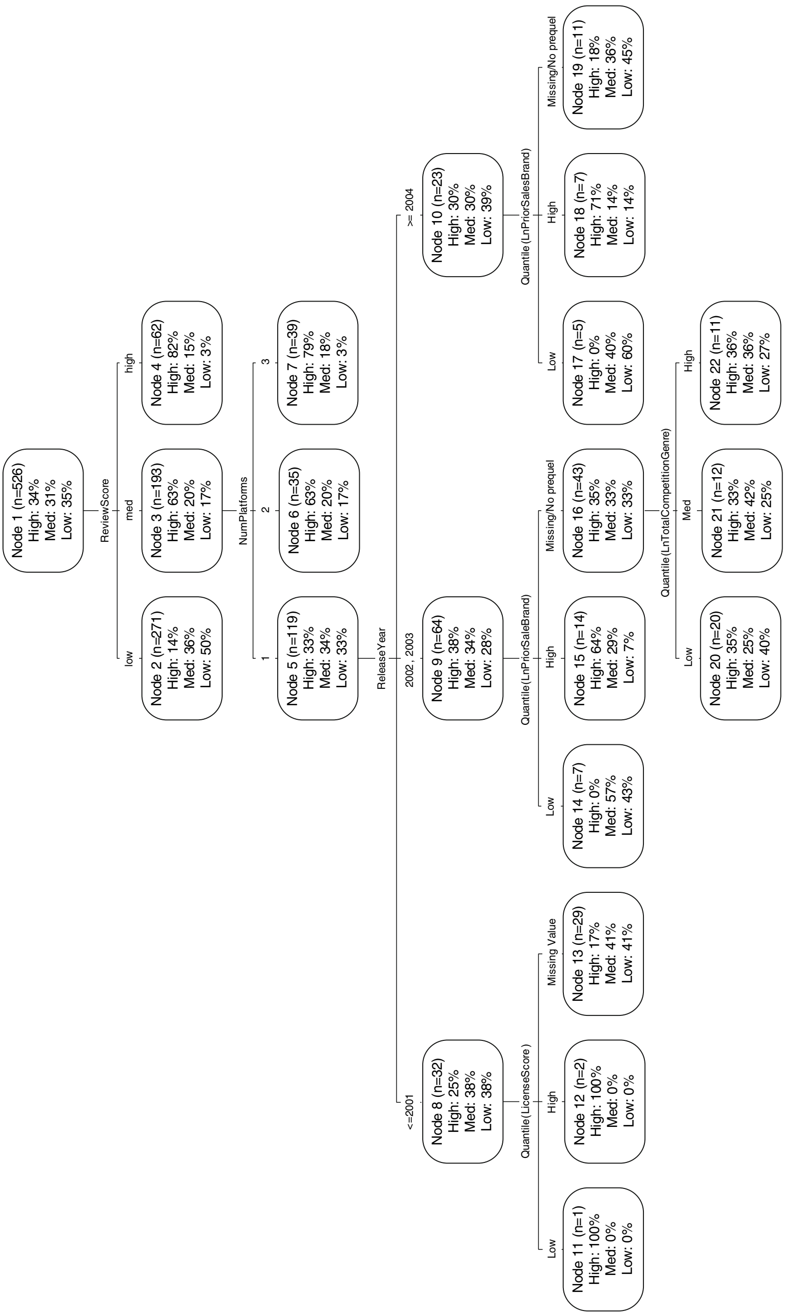

Fig. 6 The classification tree used to predict video game sales

no information gain relative to the a priori estimates in the tree's root node. At other nodes, however, the predictions are much less equivocal. For example, a single feature value, ReviewScore = high, increases the unsmoothed estimated probability of high sales from 0.34 to 0.82.

5.4 Node transformations

The classification tree in Fig. 6 provides examples of all three variable types described in Sect. 4. Starting at the root node, the ReviewScore variable is a semi-controllable variable—it is a non-deterministic outcome of some other decision or decisions made earlier in the development process. These antecedents (or proxy variables) must be identified in order for ReviewScore to be useful to decision makers. According to our contacts at GameCorp, the best single predictor of a game's review score is development effort, which is a controllable variable. We created a proxy variable called DevEffort to condition the probability distributions of ReviewScore, as shown in the decision subtree in Fig. 7. Since the industry-level data used for data mining tells us nothing about the firm-specific relationship between development cost and review scores, we had to seek this information within GameCorp. The first step was to ask a panel of seven developers and producers employed by the firm to define three levels of development effort (low, med, high) and to assign representative costs to each level. We then asked the members of the panel to consider 265 existing video game titles (varying in genre, release date, developer, and so on) and assign a development cost level to each game. Note that the panel members were not asked to estimate the development cost of each game (the majority of which were developed by other firms) but rather to estimate the development cost of the game if GameCorp had been the developer. Just over 100 games (41%) were assigned a cost category by all panel members and 83% of the games had at least one estimate.

According to the average cost estimates provided by the panel, high development effort costs GameCorp roughly \$17.14 M.[4] The average of the panel's estimate of med and low development costs in contrast were \$7.79 M and \$3.86 M, respectively. These estimates were used as costs along the decision arcs emanating from the DevEffort decision node. The panel's median estimate of DevEffort for each game was then combined with the game's ReviewScore to estimate the conditional probabilities for each level of development effort.

The conditional probabilities, which are shown along the chance arcs in Fig. 7, are broadly consistent with our prior expectations: higher development effort leads, on average, to a higher probability of better review scores. We performed a regression analysis to determine the extent to which the panel's estimates of development effort predicted review score. Although the coefficient for DevEffort was highly significant, the panel's median DevEffort rating explains only 28% of the variance in ReviewScore. This low R^2 value suggests that additional explanatory variables, such as Genre or NumPlatforms should be included in the ReviewScore subtree. However, it is important to keep in mind that each additional proxy variable has an impact on the

[4] All firm-specific data has been disguised.

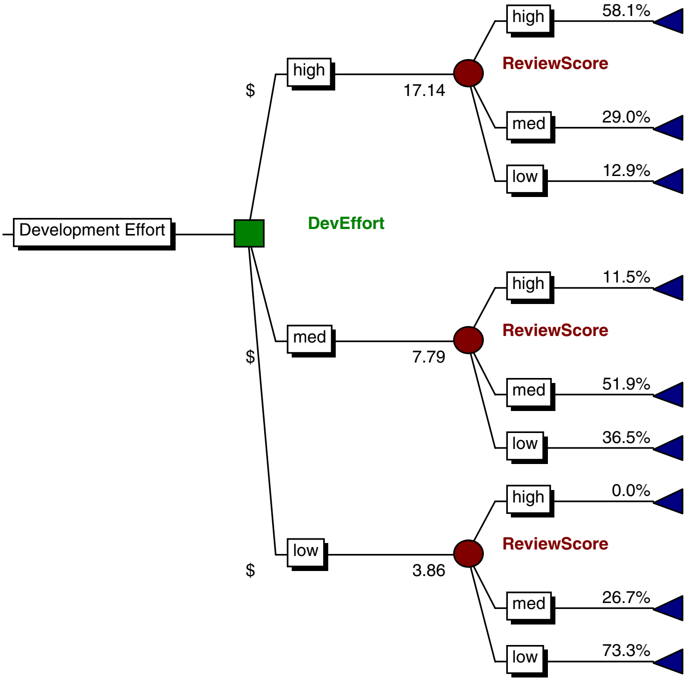

Fig. 7 The ReviewScore node in the classification tree is transformed into a decision subtree with costs along the decision arcs and conditional probabilities along the chance arcs

size of the final decision tree. A balance must be struck between predictive accuracy and decision tree complexity.

Moving down the classification tree in Fig. 6, Node 3 is split on the NumPlatforms variable. Over 65% of the games in our sample released from 2000 to 2004 were developed for a single platform while 15% were developed for all three major consoles. Although developers have long ported successful games to other consoles, an industry trend is to develop versions of the game for all three consoles simultaneously. Since the value of the NumPlatforms variable is fully controllable, we transformed the NumPlatforms classification tree node into a decision node with the same name. The only source of uncertainty in the transformation concerns the actions costs along each decision arc, which must be estimated from other sources.

Unfortunately, reliable estimates of porting costs were not available from our contacts at GameCorp. We therefore assigned $0 as the cost for NumPlatforms = 1 and made very rough estimates of the incremental costs of two and three platforms. In reality, the cost of porting a game depends on its genre. *Ceteris paribus*, it is cheaper to port a game in which art, music, and level design are major cost drivers since these assets require little or no modification from platform to platform. In addition, the cost of porting a game may depend on architectural and infrastructural decisions made by the firm. Some development firms have adopted middleware technology to automate some of the more onerous porting tasks (Reimer 2005).

The next split in the classification tree occurs at Node 5 on the ReleaseYear variable. The inclusion of ReleaseYear in the classification tree indicates that in some circumstances the factors that impact video game sales are time-dependent. For example, the LicenseScore variable occurs under the ReleaseYear $\leq$ 2001 branch but does not appear under the ReleaseYear $\geq$ 2004 branch. This suggests that the importance of licenses in predicting sales has changed over time. It is clear, however, that ReleaseYear does not fit within the controllable/uncontrollable typology introduced in Sect. 4. Instead, it is analogous to a control variable in regression: it extracts some of the variance attributable to time-dependent trends from the training data and thereby increases the accuracy of the lower-level branches. Since the point of the decision analysis is to make predictions about future games, only the ReleaseYear $\geq$ 2004 branch is relevant to the transformation. The irrelevant branches were pruned by replacing Node 5 in the classification tree with Node 10.

As an aside, it is instructive to consider the transformations that would be involved if variables on the irrelevant branches were found elsewhere in the classification tree. Recall that Quantile(LicenseScore) = {low, med, high} represents the "attractiveness" of a licensed brand as assessed by a sample of consumers. According to the subtree rooted at Node 8, the decision whether to use a high quality license, a lower quality license, or no license can have an impact on game sales. When considering whether to license different brands for use in a new video game, the developer could use survey methods to estimate each brand's LicenseScore. However, each license entails different upfront and ongoing costs and these costs would have to be added to the decision arcs emanating from the LicenseScore decision node. The splitting variable at Node 16, Quantile(LnTotalCompetitionGenre), is a transformed measure for game i of the number of games for which Genre(j) = Genre(i) and ReleaseDate$(j) \leq$ ReleaseDate(i), $\forall j \neq i$. Since competition depends in large part on the actions of other developers and is therefore uncontrollable, Node 16 would be replaced by a chance node representing the probability distribution of different levels of competition encountered on the game's release. The probabilities along each arc would be estimated from the relative frequencies in the classification tree: $p(\text{low}) = 20/(20+12+11) = 0.47$, $p(\text{med}) = 0.28$, and $p(\text{high}) = 0.26$.

Returning to the relevant (forward-looking) part of the classification tree, recall that the PriorSalesBrand variable used to split Node 10 represents the sales of all previously released games within the game's brand franchise. When *Madden NFL Football 2005* was released, for example, it was extending a franchise that had already earned more than \$480 M. The PriorSaleBrand for the game (the sales of the most recent prequel, *Madden NFL Football 2004*) was over \$202 M. We chose to model such variables as uncontrollable even though, as is discussed in the section on real options below, prequel sales is an example of a *path dependent* semi-controllable variable.

5.5 An optimal development policy

The transformation of the classification tree resulted in a decision tree with 94 nodes, 63 of which were leaf nodes with payoff estimates. The decision tree was constructed and rolled back using commercial decision tree software and Fig. 8 shows the final

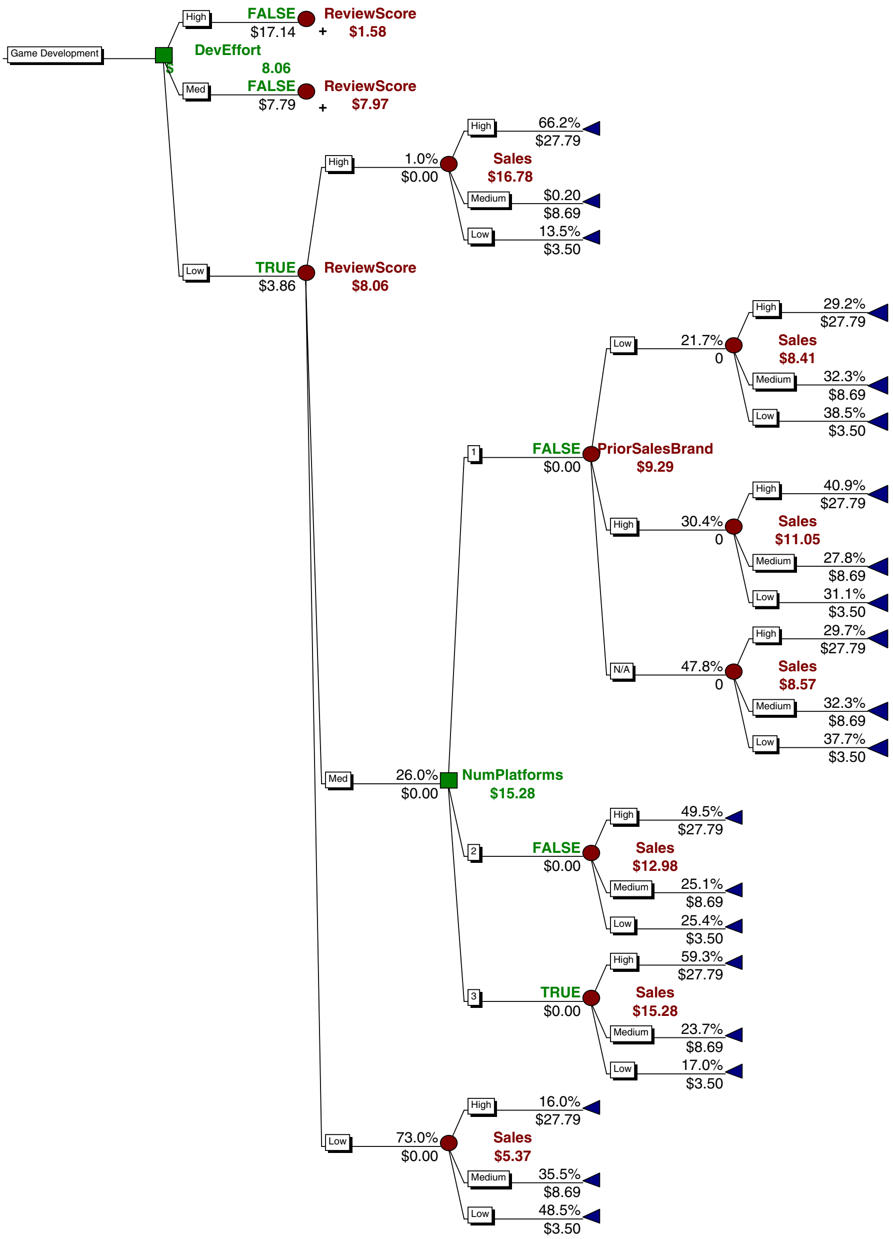

Fig. 8 A portion of the decision tree showing the optimal policy for GameCorp

result. Since GameCorp is a large firm relative to the size of the development investments, we maximized risk-neutral expected monetary value. Starting at the root node on the left of the tree in Fig. 8, the first decision that GameCorp faces is its level of development effort in a new game. The topmost branch—which corresponds to high development effort—has a net expected value of \$1.58 M. The decision arc has been labeled "FALSE" by the rollback algorithm to indicate that it is not optimal and all subsequent nodes along the path have been collapsed to a single node to save space in the figure. The center node, which corresponds to medium development effort, has an expected value of 7.97 M and has also been designated suboptimal and collapsed. The bottom branch, which corresponds to the lowest development effort, has an expected value of \$8.06 M, which is the highest among the three decision arcs (hence the "TRUE" designation). The next node along the optimal path is a chance node corresponding to ReviewScore. As discussed previously, the probability distribution of review outcomes is conditioned on development effort. If ReviewScore = high, the next node is the terminal chance node for LTDSales. Based on information provided by the classification tree (which has been smoothed using the procedure described in Zadrozny and Elkan (2001)), a game with a high review score has a higher probability of achieving high sales. The expected payoff from high sales is \$27.79 M whereas the expected payoff from low sales is a mere \$3.5 M.

According to the classification tree, NumPlatforms has significant predictive power if and only if ReviewScore = med. This conditional dependency is shown in Fig. 8 as an asymmetrical branch in the decision tree. We have set the action costs along the different decision arcs to zero in order to more clearly show the revenue implications of porting games. According to the decision tree, the expected revenue from releasing a game with a medium review score on a single platform is \$9.29 M. Expected revenues increase to \$12.98 M and \$15.28 M for two and three platforms, respectively. The decision whether to port a game to additional platforms therefore depends on whether porting costs are less than the incremental increases in expected revenues.

Generally, the implications of asymmetrical branches depend on whether they contain chance or decision nodes. Since NumPlatforms is a decision node, the rational action along the ReviewScore = high and low branches is to release the game on a single platform. This is because, according to analysis of historical data, the number of platforms on which a game is released has no discernable impact on game sales. However, the action costs associated with porting games are predictable and most likely significant. Of course, this reasoning is predicated on the assumption that review scores are known early enough in the development process to influence porting decisions. If not, either the decision tree has to be redrawn with the NumPlatforms decision to the left of the ReviewScore chance node or the firm has to invest in an early proxy for reviews, such as focus group testing, and incorporate the results into the decision tree.

Overall, the most surprising result of the decision tree analysis is that low development effort is recommended over higher development effort. This occurs even though the probability of a high review score is significantly higher with high development effort and, as the classification tree shows, a high review score is the best predictor of high sales. According to this probabilistic cost-benefit analysis and the firm-specific information it contains, the costs and risks associated with the industry's prevailing focus on expensive blockbusters are too high for GameCorp. The firm's optimal strategy is to invest less than $4 M per game and, depending on porting costs, target multiple consoles.

6 The problem of path dependencies

As noted above, an important shortcoming of the decision tree model in Fig. 8 is that it treats PriorSalesBrand as a chance variable. The power of an established brand is well recognized within the video game industry. In 2005, Electronic Arts published one new game and 25 sequels to its existing games (Reimer 2005). However, the accumulated value of a brand franchise is clearly not an uncontrollable variable. It is attributable in large part to prior development decisions by the firm.

The impact of the sales of game i on the sales of game j is significant because it means that decisions regarding game j are highly path dependent. When evaluating the expected value of game i, the decision maker must include the possibility that the game is highly successful and provides the firm with the option of following on with one or more sequels (which, according to historical data are significantly more likely than average to also be successful). The decision tree in Fig. 8 underestimates the value of high sales because it does not consider the path dependencies that have been shown empirically to exist between games of the same brand.

This type of interdependency between decisions over time corresponds to a "real option" in the decision making literature. Table 1 illustrates the structural similarities between a financial option and the analogous real option created in the video game context by sequels.

One means of incorporating the value of real options into the methodology presented in this paper is to model a development portfolio of games instead of evaluating each game individually. This approach is shown at a very high level of detail in Fig. 9. The problem is that a two-game portfolio would require the entire decision tree for Game 2 to be grafted on to the 63 leaf nodes of the decision tree for Game 1. A three-game

Table 1 A comparison between the decision stages of a financial option and a real option

Stage in the sequence	Financial option	Real option: video game sequel
Initial investment decision: buy or do not buy a call option	Whether to purchase an option to buy a share of IBM stock on a particular expiry date at a particular strike price	Whether to invest heavily in game i in order to increase the probability of it being successful
Resolution of uncertainty	On the option's expiry date, the market price of IBM stock is known with certainty	When the time comes to invest in a new game j, the market performance of game i is known
Second investment decision: exercise or abandon the call option	Exercise the option if IBM's market price is greater than the option's strike price; abandon the option otherwise	Make game j a sequel of game i if i was successful; otherwise, develop a game unrelated to i

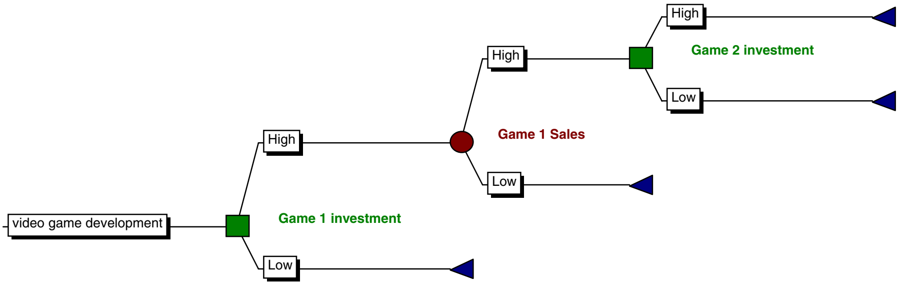

Fig. 9 A high-level decision tree showing a real options sequence in which Game 2 (a sequel) may be made if Game 1 is successful

portfolio would require a decision tree with a total of $63^3 = 250,047$ leaf nodes, and so on.

The fundamental difficulty created by the existence of real options in a decision problem is that the decision maker must know what decisions will be made in the future in order to make a decision in the present. Specifically, the precise circumstances in which the option will be exercised must be known in advance in order to evaluate its value. The implication is that determining the value of a real option presupposes the solution of a stochastic planning problem. Unfortunately, the representations of such problems tend to grow exponentially in the size and the complexity of the problem (Boutilier et al. 1999). This "curse of dimensionality" occurs not because of the shortcomings of the decision tree representation, but because the inherent combinatorial complexity of planning over a large number of contingencies.

A literature addressing different techniques for coping with the complexity of real options is well established (e.g., Trigeorgis 1996; Smith and McCardle 1999; Copeland and Tufano 2004; Luehrman 1998). Although a complete evaluation of real options pricing techniques is beyond the scope of this paper, many of the most popular approaches in the literature are based on financial option pricing models. Such models avoid the curse of dimensionality by making restrictive assumptions about the effects of uncertainty over time and thereby avoid the requirement to explicitly model all possible contingencies. Unfortunately, the restrictive assumptions may not hold in the strategic decision making context considered in this research (see e.g., Lander and Pinches 1998; Brydon 2006). The difficulties created by real options in data mining/decision making integration thus remain largely unresolved.

7 Conclusions, limitations, and implications for practice

The objectives of this paper were to present a methodology for converting classification trees into decision-theoretic cost-benefit models and to assess the feasibility of applying the approach to a large, complex, real-world decision problem. We used off-the-shelf data mining software and commercially available market research data to create a predictive model of video games sales. Our experience in this case study showed that classification trees, by themselves, are of little use to decision makers. The explanatory variables in classification trees are often the direct or indirect results of managerial decisions and these decisions have cost implications. Selecting an optimal path through a classification tree requires a full accounting of the costs in addition to the benefits predicted by the tree.

Decision trees (as the term is used in decision analysis) provide a principled means of making cost-benefit decisions in the face of uncertainty. The methodology presented in the paper provides a set of techniques for transforming classification trees into decision trees. Interestingly, the results of the video game example provide a perspective on blockbusters and sequels that contradicts the prevailing view held by many executives in the video game industry. Although it is impossible in such a complex environment to evaluate the quality of the resulting policy against alternative development policies, we note that some of our contacts at GameCorp left the firm during the course of our research to found a new game studio. According to the new firm's website, it seeks to create games with "the highest density of fun per development dollar". This development philosophy of course falls short of validation; however, it indicates that some executives in the industry already believe what the data suggests to us—a game can be an economic success without being an expensive blockbuster.

From the perspective of our research objectives, the case illustrated that decision theoretic feedforward control can be feasible in a complex decision environment. Despite a large number of candidate explanatory variables, the data mining algorithm provided a compact predictive model and enabled a final decision tree of a manageable size. This suggests that the methodology presented in this research can be applied to other challenging decision making environments.

The primary limitation of this research is that it is preliminary and exploratory. Given our focus on developing a methodology and broadly assessing its feasibility, we have made only preliminary efforts to start with the best possible classification/probability estimation tree or to assess the stability of the tree. Part of the problem is that the data set used for training—despite being the best available in the industry—has several significant shortcomings. In a separate multiple regression analysis of the data, we were unable to explain more than half of the observed variance in LTDSales using the explanatory variables in the data set. Variables that are conspicuously absent from the data set, such as those concerning the marketing and promotion of games, are likely required to achieve a more accurate predictive model. Ideally, this stream of research will show firms in the video game industry how data can be used to inform development decisions and motivate them to collect more and better data about industry-level sales performance and firm-level costs.

A second limitation concerns our use classification trees for probability estimation. Several researchers, such as Zadrozny and Elkan (2001) and Provost and Domingos (2003) have argued that probability estimation is sufficiently distinct from classification to require its own set of techniques and algorithms. For example, the common practice of validation pruning is identified as a source of bias in probability estimation. Although we disabled pruning when constructing our classification tree, we accepted Enterprise Miner's defaults for a minimum number of training examples in each node. The impact was much the same as pruning in that our final tree contains only five levels of branching and 15 leaf nodes. An unconstrained version of the same tree contains up to ten levels of branching and 48 leaf nodes. Although the increase in the size of the unconstrained tree is not so large as to threaten the feasibility of the approach, it does represent a significant increase in the amount of work required to construct and parameterize a decision tree. More research is required to identify the best trade-off between accuracy and modeling effort when building probability estimation trees for decision analysis.

Finally, the complexity challenges created by real options need to be addressed. Decision trees have been shown to be impractical in many decision making domains in which there are multiple sequential decisions and strong path dependencies, such as new product development in the pharmaceuticals industry (Brydon 2006) and resource extraction in the oil and gas industry (Smith and McCardle 1999). Feature-based alternatives to decision trees, such as *structured* dynamic programming (Boutilier et al. 1999) may provide a more scalable alternative to conventional decision-analytic techniques.

Acknowledgements The authors thank the Natural Science and Engineering Research Council of Canada (NSERC) and the Social Sciences and Humanities Resource Council of Canada (SSHRC) Initiatives for the New Economy (INE) for financial assistance.

References

Apte C, Li B, Pednault E, Smyth P (2002) Business applications of data mining. Commun ACM 45(8): 49–53

Barry E, Kemerer C, Slaughter S (2006) Environmental volatility development decision and software volatility: a longitudinal analysis. Manage Sci 52(3):448–464

Boutilier C, Dean T, Hanks S (1999) Decision-theoretic planning: structural assumptions and computational leverage. J Artif Intell Res 11:1–94

Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and regression trees. Wadsworth International Group, Belmont

Brooks RA (1991) Intelligence without reason. In: Proceedings of the 12th international joint conference on artificial intelligence, pp 569–595

Brydon M (2006) Evaluating strategic options using decision-theoretic planning. Inform Technol Manage 7(1):35–49

Charne J (2006) Understanding and negotiating termination issues in video game development contracts. Comput Internet Lawyer 23(9):27–33

Clemen R (1996) Making hard decisions: an introduction to decision analysis. Duxbury Press, North Scituate

Copeland T, Tufano P (2004) A real-world way to manage real options. Harv Bus Rev 82(3):90–99

Davenport T, Harris J, De Long D, Jacobson A (2001) Data to knowledge to results: building an analytic capability. Calif Manage Rev 43(2):117–138

Gaume N (2006) Nicolas Gaume's views on the video games sector. Eur Manage J 24(4):299–309

Haughton D, Deichmann J, Eshghi A, Sayek S, Teebagy N, Topi H (2003) A review of software packages for data mining. Am Stat 57(4):290

Holloway C (1979) Decision making under uncertainty: models and choices. Prentice-Hall, Englewood Cliffs

Keeney R, Raiffa H (1976) Decisions with multiple objectives: preferences and value tradeoffs. Wiley, London

Khatri N, Ng HA (2000) The role of intuition in strategic decision making. Hum Relat 53(1):57–86

Kolodny L (2006) Global video game market set to explode. BusinessWeek. http://www.businessweekcom/innovate/content/jun2006/id20060623_163211htm. Accessed 23 June 2006

Lander DM, Pinches GE (1998) Challenges to the practical implementation of modeling and valuing real options. Q Rev Econ Financ 38(4):537

Luehrman TA (1998) Strategy as a portfolio of real options. Harv Bus Rev 76(5):89–99

McCarthy I, Tsinopoulos C, Allen P, Rose-Anderssen C (2006) New product development as a complex adaptive system of decisions. J Prod Innovat Manag 23(5):437–456

Melville P, Provost F, Saar-Tsechansky M, Mooney R (2005) Economical active feature-value acquisition through expected utility estimation. In: Proceedings of the 1st international workshop on utility-based data mining (UBDM '05), pp 10–16

Pednault E, Abe N, Zadrozny B, Wang H, Fan W, Apte C (2002) Sequential cost-sensitive decision making with reinforcement learning. In: Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining

Pham A, Friedman J (2007) Microsoft's 'Halo 3' expected to ring up giant sales. Los Angeles Times, September 24. http://www.latimes.com/entertainment/news/newmedia/la-fi-halo24sep24,0,701643.story?coll=la-home-entertainment. Accessed 10 Oct 2007

Provost F, Domingos P (2003) Tree induction for probability-based ranking. Mach Learn 52(3):199–215

Pyle D (1999) Data preparation for data mining. Morgan Kaufmann Publishers, Los Altos

Raiffa H (1968) Decision analysis: introductory lectures on choices under uncertainty. Addison-Wesley, Reading

Rahul A (2006) Crossing the analytics chasm. Bus Intell J 11:1. http://www.tdwiorg/Publications/BIJournal/displayaspx?ID=7892. Accessed 11 Sep 2006

Reimer J (2005) Cross-platform game development and the next generation of consoles ars technica. http://arstechnicacom/articles/paedia/hardware/crossplatformars. Accessed 18 Nov 2006

Russell S, Norvig P (1995) Artificial intelligence: a modern approach. Prentice Hall, Englewood Cliffs

Schocken S, Jones C (1993) Reframing decision problems: a graph-grammar approach. Inform Syst Res 4(1):55–87

Simon HA (1977) The new science of management decision. Prentice Hall, Englewood Cliffs

Smith JE, McCardle KF (1999) Options in the real world: lessons learned in evaluating oil and gas investments. Oper Res 47(1):1–15

Trigeorgis L (1996) Real options: managerial flexibility and strategy in resource allocation. MIT Press, Cambridge

Walfisz M, Zackariasson P, Wilson T (2006) Real-time strategy: evolutionary game development. Bus Horizons 49(6):487–498

Zadrozny B, Elkan C (2001) Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: Proceedings of the 18th international conference on machine learning, pp 609–616

Zhang K, Fan W, Buckles B, Yuan X, Xu Z (2006) Discovering unrevealed properties of probability estimation trees: on algorithm selection and performance explanation. In: Proceedings of the sixth international conference on data mining (ICDM '06), pp 741–752