Yale University
Department of Computer Science

Decision Making in the Presence of Noise

Michael J. Fischer Sophia A. Paleologou

YALEU/DCS/TR-875
October 1991

This research was supported in part by National Science Foundation grant IRI–9015570.

Decision Making in the Presence of Noise*

Michael J. Fischer Sophia A. Paleologou
Department of Computer Science
Yale University

Abstract

We consider problems of decision making based on imperfect information. We derive Bayesian optimal decision procedures for some simple one-person games on trees in which the player is given redundant but noisy information about the true configuration of the game. Our procedures are computationally efficient, and the decision rules which they implement are describable by simple formulas. Not surprisingly, the presence of noise greatly affects the decision procedure, and decisions procedures that are optimal for the corresponding noiseless games may be far from optimal in the presence of noise. In many cases, the optimal decision depends not only on the given noisy data but also on knowledge of the expected amount of noise present in the data. For arbitrary $m \in \mathcal{N}$, we present examples in which the optimal decision changes m times as the probability of error in an individual datum increases from 0 to 1/2. Thus, no decision procedure that is insensitive to (or does not know) the amount of uncertainty in the data can perform as well as one that is aware of the unreliability of its data.

1 Introduction

Many complex real-life situations require that people make decisions based on imperfect information. Widely used algorithms for decision making in such complex environments often overlook the fact that the information with which they are provided is unreliable and use this information as if it were accurate, hoping that this will nevertheless lead to a good decision.

A typical example of such an algorithm is the Shannon chess playing algorithm, which looks k levels ahead in the game tree, evaluates the strength of each resulting board position, and then uses standard "min-max" techniques to choose the most promising next move. If the evaluations were 100% accurate, this would lead to optimal play, but it is unclear how good a move this produces in real chess programs. In a recent conference for Learning, Rationality, and Games at the Santa Fe Institute, John Geanakoplos and Larry Gray [GG91b] gave examples of simple one-person games in which the Shannon algorithm was provably non-optimal and in which its performance actually deteriorated as the amount of permitted look-ahead (and hence the amount of data upon which to base one's decision) increased.[1]

In this paper, we investigate the structure of the Bayesian optimal decision in the simple games of Geanakoplos and Gray. Rather surprisingly, the optimal decision can be expressed

*This research was supported in part by National Science Foundation grant IRI–9015570.

[1] Judea Pearl has also noted such phenomena in chess and given probabilistic game models in which reaching deeper consistently degrades the quality of the Shannon algorithm's decision [Pea83].

by compact, closed-form formulas of low computational complexity. From these formulas, we gain qualitative insights into the Bayesian optimal decision. We observe, for example, that no algorithm that bases its decision solely on the information contained at the level-k nodes of the tree (as the Shannon algorithm does) is Bayesian optimal. We also give an example to show that the optimal decision sometimes depends not only on the information contained at the nodes, but also on knowledge of the expected amount of noise present in the data. Thus, an algorithm that is aware of the imperfection of its data can do better than one that is not.

It is tempting to apply these insights to chess in order to obtain improved algorithms, and we are hopeful that workers on chess will find our results enlightening. Nevertheless, we should point out a number of important differences between our games and the problem of playing chess. In our games, uncertainty arises from two underlying sources of randomness. The instance of the game to be played is chosen at random (as in card games such as bridge or poker), and the information about the chosen game that is given to the player is also chosen at random. Thus, a player is presented with probabilistic, partial information about the true underlying game. The difficulty the player faces is in knowing which game is being played, not how to play the game once it is known. In chess, on the other hand, the underlying game tree is fixed, and the player's information is computed by a deterministic procedure of low computational complexity. The barrier to optimal play is not the lack of accurate information but the apparent intractability of the computational problem of making good use of that information. An important research problem is to clarify the relationship between probabilistic and computational sources of uncertainty in decision problems.

2 A Basis of Tree Games

In this section, we provide the notation and definitions we will be using throughout the paper.

For any complete binary tree T, $nodes(T)$ and $leaves(T)$ are the sets of nodes and leaves of T respectively. For notational convenience, we often identify T with $nodes(T)$ and use $x \in T$ to mean $x \in nodes(T)$. If $x \in nodes(T) - leaves(T)$, then $Lchild(x)$ and $Rchild(x)$ denote the left and right children of x respectively. Whenever we consider subtrees of T, we restrict ourselves to subtrees that satisfy the following property: if $x \in nodes(T)$ is the root of the subtree X, then all the descendants of x are also in $nodes(X)$. Similarly, whenever we consider paths in T, we restrict ourselves to paths from the root of T to a leaf; $paths(T)$ is the set of all such paths in T. Finally, if $\pi \in paths(T)$, $nodes(\pi)$ is the set of all nodes of T on path π.

Definition 1 Let T be a complete binary tree. A function $\lambda : nodes(T) \rightarrow \{0,1\}$ is called a *labelling* of T, and the tuple (T,λ) is called a *labelled tree.* For $x \in nodes(T)$, $\lambda(x)$ is then called the *label* of node x under λ.

If λ is a labelling of T, then λ_X is the restriction of λ to the nodes of X; i.e., $\lambda_X = \lambda|_{nodes(X)}$. Obviously, $\lambda_T = \lambda$. Also, if r is a single node of T, we sometimes write λ_r to denote $\lambda|\{r\}$.

Definition 2 Let T be a complete binary tree. A labelling λ of T is called *proper* iff it satisfies the *max-property*; that is, for all $x \in nodes(T) - leaves(T)$,

$$\lambda(x) = \max\{\lambda(Lchild(x)), \lambda(Rchild(x))\}.$$

The tuple (T, λ) is then called a *MaxTree.*

Definition 3 Let T be a complete binary tree. A proper labelling λ of T is called *winning* iff there exists $\alpha \in \mathit{leaves}(T)$ such that $\lambda(\alpha) = 1$. A proper labelling that is not winning is called *losing.*

Definition 4 Let (T, λ) be a MaxTree. A path $\pi \in \mathit{paths}(T)$ is called *winning* iff $\lambda(x) = 1$ for all $x \in \mathit{nodes}(\pi)$. Similarly, a leaf $\alpha \in \mathit{leaves}(T)$ is called *winning* iff $\lambda(\alpha) = 1$.

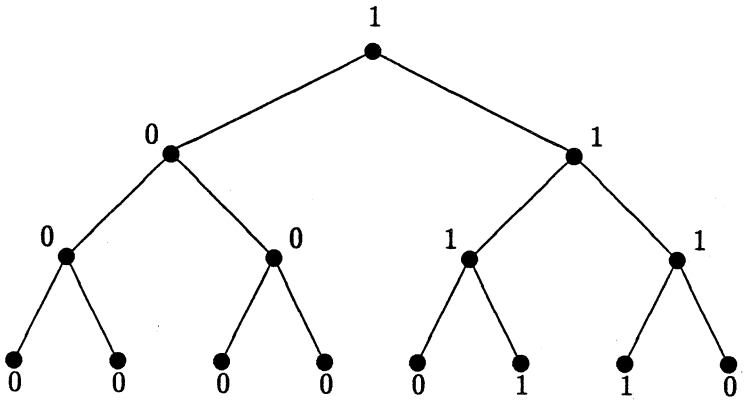

Figure 1: A MaxTree with two winning leaves.

As a result of the max-property, any proper labelling λ can be uniquely specified by the labels it assigns to the leaves of T; the labels of all the internal nodes of T can then be recursively computed as the maximum of the labels of their children. This recursive node-labelling process is widely known in game theory as *backward induction* (see [Shu82]). An example of a MaxTree T of uniform depth $k = 3$ is given in Figure 1.

In this paper, we generally consider MaxTrees with *exactly one* leaf labelled 1 and all other leaves labelled 0. For every $\alpha \in \mathit{leaves}(T)$, λ^α denotes the proper labelling of T which assigns the label 1 to α and the label 0 to all other leaves of T; i.e., $\lambda^\alpha(\alpha) = 1$, and $\lambda^\alpha(\beta) = 0$ for all $\beta \in \mathit{leaves}(T) - \{\alpha\}$.

MaxTrees can be thought of as game trees for natural one-person games which model multistage decision processes. Each node of the tree models a state of the game. At every internal node, the unique player of the game is faced with two alternatives, going Left or going Right, while the leaves of the tree represent final states of the game. They can be used as a basis for defining a variety of related simple (single-stage) one-person games. An example of such a simple game follows:

Example 1 Let (T, λ) be a MaxTree. Given (T, λ), the player is asked to choose a path from the root of T to a leaf. If the path she chooses leads to a leaf labelled 1 under λ, the player wins; otherwise, she loses.

An obvious way for the player to play the game of example 1 is to choose Left and Right according to which of the two children of the current node is labelled 1. This strategy will cause the player to "walk" down a path labelled with 1's and to reach a winning leaf after k

moves, where k is the depth of T. In the special case where all nodes of the given MaxTree are labelled 0, the player always loses.

The game of example 1 is not interesting, since it provides the player with complete and accurate information, thereby turning her decision-making into a trivial process. In example 2 below, we present a variant of that game where the information visible to the player has been corrupted by noise.

Before we proceed with the necessary definitions, we introduce the following notational conventions: if Z is a random variable, we identify Z with the corresponding random experiment and use z to denote the outcome of this experiment, sometimes also referred to as the realization of the random variable Z. Furthermore, whenever there are no grounds for confusion, we use $\mathbf{prob}[z]$ to denote $\mathbf{prob}[Z = z]$, the probability that $Z = z$.

Definition 5 Let $p \in (0, 1/2)$ be a constant.[2] A 0/1 random variable Z is called a *random coin with bias p* iff

$$\mathbf{prob}[Z = 0] = 1 - p \text{ and } \mathbf{prob}[Z = 1] = p.$$

Definition 6 Let $p \in (0, 1/2)$ be a constant. Let $\{Z_x : x \in nodes(T)\}$ be a collection of independent random coins with the same bias p. Let (T, λ) be a MaxTree and let $x \in nodes(T)$. A 0/1 random variable V_x is called a *random corruption of the label* $\lambda(x)$ *with error probability p* iff

$$V_x = \lambda(x) \oplus Z_x.$$

According to definition 6, the random corruptions of the labels of the nodes in T satisfy the following two properties:

- *locality*: for all $x \in nodes(T)$, the corruption V_x depends only on the label $\lambda(x)$ and is independent of the labels $\lambda(y)$ of all $y \in nodes(T) - \{x\}$;
- *independence*: for all $x, y \in nodes(T)$ with $x \neq y$, the corruptions V_x and V_y of the labels $\lambda(x)$ and $\lambda(y)$ are independent random variables.

Definition 7 Let (T, λ) be a MaxTree. If, for every node x in T, v_x is the outcome of a random corruption V_x of its label $\lambda(x)$ with error probability p, then the labelling $\theta : nodes(T) \to \{0, 1\}$, such that $\theta(x) = v_x$, is called a *corrupted view* of the proper labelling λ with error probability p.

We refer to the labels of the nodes of T under λ as *actual labels*, while we refer to the labels of the nodes of T under θ as *corrupted* or *observed labels*. Unlike λ, θ does not necessarily satisfy the max-property of proper labellings.

Example 2 Let (T, Λ) be a MaxTree with Λ a random labelling of T following a probability distribution $\mathcal{P}$. Let λ be the outcome of Λ and let θ be a corrupted view of λ with error probability p. Given (T, θ, p) and the probability distribution $\mathcal{P}$, the player is asked to choose a path from the root of T to a leaf. If the path she chooses ends in a leaf labelled 1 under λ, the player wins; otherwise, she loses.

[2] If $p = 1/2$, the corruption V_x is independent of the labelling $\lambda(x)$. Also, if $p > 1/2$, the player can use the corrupted view θ to construct a new labelling θ' by taking $\theta'(x) = 1 \oplus \theta(x)$. We can think of θ' as another corrupted view of λ with error probability $p' = 1 - p < 1/2$. Thus, it is reasonable for us to focus on the case $p \in (0, 1/2)$, since all other cases are either not interesting or can be reduced to the case $p \in (0, 1/2)$.

In general, no algorithm for the game of example 2 can guarantee the player a win, since she has access only to the corrupted labels of the nodes in T. In the absence of an algorithm that guarantees success, the player might alternatively look for an algorithm that maximizes her chances of winning, thereby exploiting the probabilistic structure of the game under consideration. She can use the view θ and the error probability p to update her prior knowledge of the distribution $\mathcal{P}$ and choose the leaf that is most likely to be winning, given the data visible to her (see section 4). An algorithm that computes the decision with the maximum probability of winning, given all available information, is *Bayesian optimal.*

In this paper, we present and analyze two simple games defined on MaxTrees, the second of which is the game of example 2 above. We show that, in many cases, seemingly intractable computations can be reduced to efficient algoritms for computing Bayesian optimal decisions. (For a general introduction to probability theory, see [Ros76].)

3 Game I: Choose a Subtree

We first consider a one-player, one-move game played on a MaxTree. The current state of the game is modelled by the root node, and the player is asked to choose one move—Left or Right, that takes her closer to a winning leaf. More formally, we have:

Game I: Let (T, Λ) be a MaxTree with Λ a random labelling following a probability distribution $\mathcal{P}$. Let λ be the outcome of Λ and let θ be a corrupted view of λ with error probability p. Given (T, θ, p) and the probability distribution $\mathcal{P}$, the player is asked to choose a subtree $Y \in \{L, R\}$, where L and R are the left and right subtrees of T. If Y contains a leaf labelled 1 under λ, the player wins; otherwise, she loses.

For the purposes of our probabilistic analysis, we fix the following probability distribution $\mathcal{P}$: for all proper labellings λ of T,

$$\mathbf{prob}[\lambda] = \begin{cases} 1/2^k & \text{if } \lambda = \lambda^\alpha \text{ for some } \alpha \in \mathit{leaves}(T) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where k is the depth of T. Thus, we assume exactly one leaf is labelled 1. However, the techniques we use in this paper to analyze Games I and II can be extended so as to handle arbitrary probability distributions (see [GG91a]).

Let X be a subtree of T and let L and R be the left and right subtrees of X. In general, the event (λ_X winning) can be viewed as the disjoint union of three events: (λ_L losing) & (λ_R winning), (λ_L winning) & (λ_R losing), and (λ_L winning) & (λ_R winning). However, in the special case of the distribution $\mathcal{P}$ that we fixed in equation 1 above, the first two of those events are equiprobable, while the third event is impossible; that is,

- $\mathbf{prob}[(\lambda_L \text{ losing}) \;\&\; (\lambda_R \text{ winning}) \mid \lambda_X \text{ winning}] = \frac{1}{2}$
- $\mathbf{prob}[(\lambda_L \text{ winning}) \;\&\; (\lambda_R \text{ losing}) \mid \lambda_X \text{ winning}] = \frac{1}{2}$
- $\mathbf{prob}[(\lambda_L \text{ winning}) \;\&\; (\lambda_R \text{ winning}) \mid \lambda_X \text{ winning}] = 0$

3.1 Probabilistic Analysis of Game I

In this section, we provide an exact probabilistic analysis of Game I. Given the corrupted view of the game tree and the a priori information about the underlying distribution of

labellings, we compute the conditional probability of winning the game for both choices Left and Right. Although this analysis might look intractable at first glance, we show how the combinatorics nicely collapse to yield compact recursive formulas that are easy to compute.

Let X be a subtree of T and define the following two quantities:

- l_X is the probability that a random corruption of a losing labelling yields in θ_X; i.e.,

$$l_X = \mathbf{prob}[\theta_X \mid \lambda_X \text{ losing}],$$

- w_X is the probability that a random corruption of a winning labelling yields in θ_X; i.e.,

$$w_X = \mathbf{prob}[\theta_X \mid \lambda_X \text{ winning}].$$

In Lemmas 1 and 2, we derive recursive formulas that allow us to compute l_X and w_X for any subtree X.

Lemma 1 *Let X be a subtree of T. Then,*

$$l_X = \begin{cases} (1-p)^{1-v}p^v & \text{if } X \text{ is a single node} \\ l_L \cdot l_R \cdot (1-p)^{1-v}p^v & \text{otherwise} \end{cases} \tag{2}$$

where L and R are the left and right subtrees of X, and $v = \theta_X(r)$ is the observed label of the root r.

Proof: Let tree X consist of a single node r with observed label $v = \theta_X(r)$, and assume λ_X is losing; i.e., $\lambda_X(r) = 0$. Then, $v = 0$ with probability $(1-p)$, while $v = 1$ with probability p. The two possibilities can be expressed in one formula as follows:

$$l_X = \mathbf{prob}[\theta_X \mid \lambda_X \text{ losing}] = (1-p)^{1-v}p^v \tag{3}$$

In the case where X is not a single node, l_X can be computed in terms of l_L and l_R.

$$\begin{aligned} l_X &= \mathbf{prob}[\theta_X \mid \lambda_X \text{ losing}] \\ &= \mathbf{prob}[\theta_L \,\&\, \theta_R \,\&\, \theta_r \mid \lambda_X \text{ losing}] \end{aligned} \tag{4}$$

The independence of the corruptions allows us to express l_X in equation 4 as the product of the conditional probabilities of the independent events θ_L, θ_R, and θ_r.

$$l_X = \mathbf{prob}[\theta_L \mid \lambda_X \text{ losing}] \cdot \mathbf{prob}[\theta_R \mid \lambda_X \text{ losing}] \cdot \mathbf{prob}[\theta_r \mid \lambda_X \text{ losing}] \tag{5}$$

Because of the locality of corruptions, the observed label of any node in X depends only on its own actual label. Furthermore, λ_X is losing if and only if all λ_L, λ_R, and λ_r are losing. Combining these observations with equation 3, we obtain:

$$\begin{aligned} l_X &= \mathbf{prob}[\theta_L \mid \lambda_L \text{ losing}] \cdot \mathbf{prob}[\theta_R \mid \lambda_R \text{ losing}] \cdot \mathbf{prob}[\theta_r \mid \lambda_r \text{ losing}] \\ &= l_L \cdot l_R \cdot (1-p)^{1-v}p^v \end{aligned}$$

■

Lemma 2 *Let X be a subtree of T. Then,*

$$w_X = \begin{cases} (1-p)^v p^{1-v} & \text{if } X \text{ is a single node} \\ \frac{1}{2}(l_L \cdot w_R + w_L \cdot l_R) \cdot (1-p)^v p^{1-v} & \text{otherwise} \end{cases} \quad (6)$$

where L and R are the left and right subtrees of X, and $v = \theta_X(r)$ is the observed label of the root r.

Proof: Let tree X consist of a single node r with observed label $v = \theta_X(r)$, and assume λ_X is winning; i.e., $\lambda_X(r) = 1$. Then, $v = 0$ with probability p, while $v = 1$ with probability $(1-p)$. The two possibilities can be expressed in one formula as follows:

$$w_X = \mathbf{prob}[\theta_X \mid \lambda_X \text{ winning}] = (1-p)^v p^{1-v} \quad (7)$$

In the case where X is not a single node, w_X can be computed in terms of l_L, l_R, w_L, and w_R.

$$\begin{aligned} w_X &= \mathbf{prob}[\theta_X \mid \lambda_X \text{ winning}] \\ &= \mathbf{prob}[\theta_L \;\&\; \theta_R \;\&\; \theta_r \mid \lambda_X \text{ winning}] \end{aligned} \quad (8)$$

The independence of the corruptions allows us to express w_X in equation 8 as the product of the conditional probabilities of the independent events $(\theta_L \;\&\; \theta_R)$ and θ_r.

$$w_X = \mathbf{prob}[\theta_L \;\&\; \theta_R \mid \lambda_X \text{ winning}] \cdot \mathbf{prob}[\theta_r \mid \lambda_X \text{ winning}] \quad (9)$$

However, under $\mathcal{P}$, the event $(\lambda_X \text{ winning})$ is the disjoint union of the equiprobable events $(\lambda_L \text{ losing}) \;\&\; (\lambda_R \text{ winning})$ and $(\lambda_L \text{ winning}) \;\&\; (\lambda_R \text{ losing})$. Equation 9 then becomes:

$$\begin{aligned} w_X &= \frac{1}{2}\Big[\, \mathbf{prob}[\theta_L \;\&\; \theta_R \mid (\lambda_L \text{ losing}) \;\&\; (\lambda_R \text{ winning})] \\ &\quad + \mathbf{prob}[\theta_L \;\&\; \theta_R \mid (\lambda_L \text{ winning}) \;\&\; (\lambda_R \text{ losing})] \,\Big] \cdot \mathbf{prob}[\theta_r \mid \lambda_X \text{ winning}] \end{aligned} \quad (10)$$

Finally, we use the independence and locality of the corruptions to further simplify equation 10:

$$\begin{aligned} w_X &= \frac{1}{2}\Big[\, \mathbf{prob}[\theta_L \mid \lambda_L \text{ losing}] \cdot \mathbf{prob}[\theta_R \mid \lambda_R \text{ winning}] \\ &\quad + \mathbf{prob}[\theta_L \mid \lambda_L \text{ winning}] \cdot \mathbf{prob}[\theta_R \mid \lambda_R \text{ losing}] \,\Big] \cdot \mathbf{prob}[\theta_r \mid \lambda_r \text{ winning}] \\ &= \frac{1}{2}(l_L \cdot w_R + w_L \cdot l_R) \cdot (1-p)^v p^{1-v} \end{aligned}$$

where $\mathbf{prob}[\theta_r \mid \lambda_r \text{ winning}]$ was substituted from equation 7. ■

In Theorem 4 below, we derive formulas that allow us to compute exactly the conditional probabilities of winning for the two choices of the player—Left and Right—based on the corrupted labelling of the game tree visible to her.

Lemma 3 *Let (T, λ) be a MaxTree and let θ be a corrupted view of λ with error probability p. Then,*

$$\mathbf{prob}[\theta \mid \lambda_L \text{ winning}] = w_L \cdot l_R \cdot (1-p)^v p^{1-v} \quad (11)$$

$$\mathbf{prob}[\theta \mid \lambda_R \text{ winning}] = l_L \cdot w_R \cdot (1-p)^v p^{1-v} \quad (12)$$

where L and R are the left and right subtrees of T, and $v = \theta(r)$ is the observed label of the root r.

Proof: We show the derivation of the formula for $\mathbf{prob}[\theta \mid \lambda_L \text{ winning}]$; the formula for $\mathbf{prob}[\theta \mid \lambda_R \text{ winning}]$ is derived similarly.

From the independence of the actual label corruptions, we have:

$$\begin{aligned}\mathbf{prob}[\theta \mid \lambda_L \text{ winning}] &= \mathbf{prob}[\theta_L \,\&\, \theta_R \,\&\, \theta_r \mid \lambda_L \text{ winning}] \\ &= \mathbf{prob}[\theta_L \mid \lambda_L \text{ winning}] \cdot \mathbf{prob}[\theta_R \mid \lambda_L \text{ winning}] \cdot \mathbf{prob}[\theta_r \mid \lambda_L \text{ winning}] \qquad (13)\end{aligned}$$

However, since λ was chosen from the probability distribution $\mathcal{P}$, λ is always winning, and λ_L is winning if and only if λ_R is losing. Combining this observation and the locality of the label corruptions, equation 13 yields:

$$\begin{aligned}&\mathbf{prob}[\theta \mid \lambda_L \text{ winning}] = \\ &= \mathbf{prob}[\theta_L \mid \lambda_L \text{ winning}] \cdot \mathbf{prob}[\theta_R \mid \lambda_R \text{ losing}] \cdot \mathbf{prob}[\theta_r \mid \lambda_r \text{ winning}] \\ &= w_L \cdot l_R \cdot (1-p)^v p^{1-v}\end{aligned}$$

■

Theorem 4 *Let (T, λ) be a MaxTree and let θ be a corrupted view of λ with error probability p. Then,*

$$\mathbf{prob}[\lambda_L \text{ winning} \mid \theta] = \frac{w_L \cdot l_R}{w_L \cdot l_R + l_L \cdot w_R} \qquad (14)$$

$$\mathbf{prob}[\lambda_R \text{ winning} \mid \theta] = \frac{w_R \cdot l_L}{w_L \cdot l_R + l_L \cdot w_R} \qquad (15)$$

where L and R are the left and right subtrees of T.

Proof: We show the derivation of the formula for $\mathbf{prob}[\lambda_L \text{ winning} \mid \theta]$; the formula for $\mathbf{prob}[\lambda_R \text{ winning} \mid \theta]$ is derived similarly.

Since λ was chosen from $\mathcal{P}$, λ is always winning, and it is equiprobable that either one of λ_L and λ_R is also winning. Thus,

$$\mathbf{prob}[\theta] = \frac{1}{2}\left(\mathbf{prob}[\theta \mid \lambda_L \text{ winning}] + \mathbf{prob}[\theta \mid \lambda_R \text{ winning}]\right) \qquad (16)$$

Using formulas 11 and 12, equation 16 becomes:

$$\mathbf{prob}[\theta] = \frac{1}{2}(w_L \cdot l_R + l_L \cdot w_R)(1-p)^v p^{1-v} \qquad (17)$$

where $v = \theta(r)$ is the observed label of the root r of T. Finally, we use *Bayes' Theorem* to combine equation 17 with formulas 11 and 12:

$$\begin{aligned}\mathbf{prob}[\lambda_L \text{ winning} \mid \theta] &= \frac{\mathbf{prob}[\theta \mid \lambda_L \text{ winning}] \cdot \mathbf{prob}[\lambda_L \text{ winning}]}{\mathbf{prob}[\theta]} \\ &= \frac{\frac{1}{2} w_L \cdot l_R \cdot (1-p)^v p^{1-v}}{\frac{1}{2}(w_L \cdot l_R + l_L \cdot w_R)(1-p)^v p^{1-v}} \\ &= \frac{w_L \cdot l_R}{w_L \cdot l_R + l_L \cdot w_R}\end{aligned}$$

■

In trying to compute **prob**[λ_L winning | θ] and **prob**[λ_R winning | θ] using equations 14 and 15, we run into computational difficulties. The quantities l_L, l_R, w_L, and w_R very quickly approach zero, so any arithmetic based on those values (using an ordinary floating-point representation) becomes impossible. However, we can rewrite those formulas in terms of the ratios w_L/l_L and w_R/l_R. It is convenient to also pull out some constants. Let $a = 2(1-p)/p$ and $c = ((1-p)/p)^2$. Let X be a k-depth subtree of T and define:

$$\Phi_X = \frac{a^{k+1}}{2} \cdot \frac{w_X}{l_X} \tag{18}$$

Lemma 5 gives a recursive formula for computing Φ_X for any subtree X.

Lemma 5 *Let X be a k-depth subtree of T. Then,*

$$\Phi_X = \begin{cases} c^v & \text{if } X \text{ is a single node} \\ (\Phi_L + \Phi_R) \cdot c^v & \text{otherwise} \end{cases} \tag{19}$$

where L and R are the left and right subtrees of X, and $v = \theta_X(r)$ is the observed label of the root r.

Proof: If X is a single node, X has depth $k = 0$, and formulas 2, 6, and 18 yield:

$$\Phi_X = \frac{a}{2} \cdot \frac{w_X}{l_X} = \frac{1-p}{p} \cdot \frac{(1-p)^v p^{1-v}}{(1-p)^{1-v} p^v} = \left(\frac{1-p}{p}\right)^{2v} = c^v$$

If X is not a single node, we can use formulas 2 and 6 to express Φ_X in terms of Φ_L and Φ_R:

$$\begin{aligned} \Phi_X &= \frac{a^{k+1}}{2} \cdot \frac{w_X}{l_X} \\ &= \frac{a^{k+1}}{2} \cdot \frac{\frac{1}{2}(l_L \cdot w_R + w_L \cdot l_R)(1-p)^v p^{1-v}}{l_L \cdot l_R \cdot (1-p)^{1-v} p^v} \end{aligned} \tag{20}$$

Taking into account that both L and R have depth $(k-1)$, equation 20 can be rewritten as:

$$\begin{aligned} \Phi_X &= \frac{a}{2} \cdot \frac{p}{1-p} \cdot \left(\frac{a^k}{2} \cdot \frac{w_L}{l_L} + \frac{a^k}{2} \cdot \frac{w_R}{l_R}\right) \cdot \left(\frac{1-p}{p}\right)^{2v} \\ &= (\Phi_L + \Phi_R) \cdot c^v \end{aligned}$$

■

The following is a restatement of Theorem 4 in terms of Φ_L and Φ_R.

Theorem 6 *Let (T, λ) be a MaxTree. Then,*

$$\textbf{prob}[\lambda_L \text{ winning} \mid \theta] = \frac{\Phi_L}{\Phi_L + \Phi_R} \tag{21}$$

$$\textbf{prob}[\lambda_R \text{ winning} \mid \theta] = \frac{\Phi_R}{\Phi_L + \Phi_R} \tag{22}$$

where L and R are the left and right subtrees of T.

Corollary 7 *Let (T, λ) be a MaxTree and let θ be a corrupted view of λ with error probability p. Then*

$$\mathbf{prob}[\lambda_L \text{ winning} \mid \theta] \geq \mathbf{prob}[\lambda_R \text{ winning} \mid \theta] \quad \textit{iff} \quad \Phi_L \geq \Phi_R \tag{23}$$

Finally, we provide the solution to equation 19, the recursive definition of Φ_X. Define the function $f: paths(T) \rightarrow \mathcal{N}$ such that, for every path π,

$$f(\pi) = \sum_{x \in nodes(\pi)} \theta(x) \tag{24}$$

Thus, $f(\pi)$ is the number of 1's among the corrupted labels of nodes on π.

Theorem 8 *Let X be a subtree of T. Then*

$$\Phi_X = \sum_{\pi \in paths(X)} c^{f(\pi)} \tag{25}$$

Proof: By induction on the depth k of X. ■

We sometimes write $\Phi_X(c)$ to emphasize the fact that Φ_X depends on c as well as on X.

3.2 Bayesian Optimal Algorithms for Game I

In this paragraph, we turn our attention to the problem of *computing* the Bayesian optimal decision for Game I. The first algorithm, A_1, shown in Figure 2, is a direct application of Corollary 7 and Lemma 5.

Theorem 9 *Let A_1 be the algorithm shown in Figure 2. Let (T, λ) be a MaxTree of n nodes and let θ be a corrupted view of λ with error probability p. On input (T, θ, p), algorithm A_1 requires $O(n)$ additions and $O(n)$ multiplications/divisions to compute the Bayesian optimal decision for Game I.*

Proof: It can be easily seen that computing Φ_L recursively from equation 19 in step 1 of algorithm A_1 requires a number of additions (multiplications) equal to the number of internal nodes in L. Similarly, computing Φ_R recursively in step 2 requires a number of additions (multiplications) equal to the number of internal nodes in R. Thus, steps 1 and 2 of algorithm A_1 require a total of $O(n)$ additions and multiplications. Step 0 of the algorithm requires only a constant number of additional operations. ■

By restructuring the computations involved in algorithm A_1 and using table look-up, we can reduce the number of multiplications/divisions to $O(\log(n))$. The resulting algorithm, A_2, shown in Figure 3, makes use of the results of Corollary 7 and Theorem 8.

Theorem 10 *Let A_2 be the algorithm shown in Figure 3. Let (T, λ) be a MaxTree of n nodes and let θ be a corrupted view of λ with error probability p. On input (T, θ, p), algorithm A_2 requires $O(n)$ additions and $O(\log(n))$ multiplications/divisions to compute the Bayesian optimal decision for Game I.*

Proof: Step 0 of algorithm A_2 takes a constant number of arithmetic operations. Step 1 requires $(\lceil \log(n) \rceil - 2) = O(\log(n))$ multiplications to compute all the powers $c^i, i = 0, 1, 2, \ldots, \lceil \log(n) \rceil - 1$. Step 2 requires at most one addition for each node of L for a total of $O(n)$ additions. Step 3 requires at most one addition for each paths $\pi \in paths(L)$ for a total of $O(n)$ additions. Step 4 takes another $O(n)$ additions. The result follows. ■

ALGORITHM A_1

Input:

T, a complete binary tree
θ, a corrupted view of T
p, the error probability of the corruption

Output:

Left or Right

Description:

Step 0:
Compute $c = ((1-p)/p)^2$.
Step 1:
Compute Φ_L recursively using equation 19.
Step 2:
Compute Φ_R recursively using equation 19.
Step 3:
Compare Φ_L and Φ_R and choose the direction—Left or Right—that corresponds to the maximum. Break ties arbitrarily.

Figure 2: A Bayesian optimal algorithm for Game I using $O(n)$ arithmetic operations.

3.3 Dependence on p

In section 3.1, we showed that the Bayesian optimal decision for Game I is a function of the corrupted view θ and the error probability p. The dependence of the optimal decision on the view θ is obvious. For example, a subtree whose corrupted view assigns the label 1 to all nodes is always preferable to a subtree whose corrupted view assigns the label 0 to all nodes.

In this section, we analyze the dependence of the optimal decision on the error probability p for a fixed corrupted view of the game tree. For the purposes of this analysis, we treat c as the independent variable. As a result, the error probability p becomes a function of c:

$$p = p(c) = \frac{1}{1+\sqrt{c}} \tag{26}$$

It is not difficult to see that $p(c)$ is a continuous and strictly decreasing function of c over $\mathcal{R}^+$. Furthermore, $c \in (1, +\infty)$ if and only if $p(c) \in (0, 1/2)$.

3.3.1 An example where knowledge of p matters

Definition 11 *Let (T, λ) be a MaxTree and let θ be a corrupted view of λ with error probability $p(c)$, $c \in (1, +\infty)$. Then the polynomial $q_T(c)$ is defined as follows:*

$$q_T(c) = \Phi_L(c) - \Phi_R(c) \tag{27}$$

where L and R are the left and right subtrees of T.

ALGORITHM A_2

Input:

T, a complete binary tree
θ, a corrupted view of T
p, the error probability of the corruption

Output:

Left or Right

Description:

Step 0:
Compute $c = ((1-p)/p)^2$
Step 1:
Compute and store in a look-up table all powers c^i,
$i = 0, 1, \ldots, (\lceil \log(n) \rceil - 1)$.
Step 2:
Compute the values of $f(\pi)$, for all $\pi \in paths(L)$, as follows: Working down from the root, compute for each node α in L the number of nodes labelled 1 by θ on the path from α to the root.
Step 3:
Add up the values of $c^{f(\pi)}$ for all paths $\pi \in paths(L)$, to compute Φ_L.
Step 4:
Repeat steps 2 and 3 substituting R for L, to compute Φ_R.
Step 5:
Compare Φ_L and Φ_R and choose the direction—Left or Right—that corresponds to the maximum. Break ties arbitrarily.

Figure 3: A Bayesian optimal algorithm for Game I using $O(\log(n))$ multiplications.

It follows from Corollary 7 that the sign of $q_T(c)$ at any point c determines the Bayesian optimal decision for the corresponding instance of Game I.

We show that there exist instances of Game I where the Bayesian optimal decision depends on p.

Theorem 12 *There exists a labelled tree (T, θ), such that the Bayesian optimal decision for Game I on input (T, θ, p) depends on p.*

Proof: Consider the labelled tree (T, θ) of Figure 4 and let L and R be the left and right subtrees of T. By inspection, L has 1 path with exactly 4 nodes labelled 1, 1 path with exactly 3 nodes labelled 1, 2 paths with exactly 2 nodes labelled 1, and 4 paths with exactly 1 node labelled 1. From equation 25, we obtain

$$\Phi_L(c) = \sum_{\pi \in paths(L)} c^{f(\pi)} = c^4 + c^3 + 2c^2 + 4c \tag{28}$$

By similar reasoning applied to R, we obtain:

$$\Phi_R(c) = \sum_{\pi \in paths(R)} c^{f(\pi)} = 4c^3 + 2c^2 + c + 1 \tag{29}$$

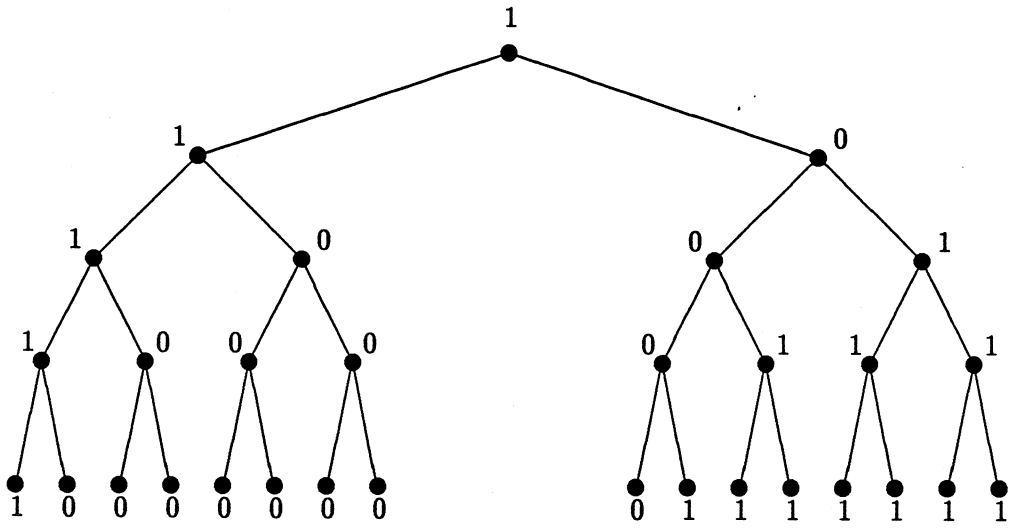

Figure 4: A view requiring knowledge of p.

Subtracting equation 29 from 28 gives:

$$\begin{aligned} q_T(c) &= \Phi_L(c) - \Phi_R(c) \\ &= c^4 - 3c^3 + 3c - 1 \\ &= (c^2 - 1)(c - \rho_1)(c - \rho_2) \end{aligned} \tag{30}$$

where

$$\rho_1 = \frac{3 - \sqrt{5}}{2} \simeq 0.382 \text{ and } \rho_2 = \frac{3 + \sqrt{5}}{2} \simeq 2.618$$

From equation 30, we see that $q_T(c)$ is strictly negative over the open interval $(1, \rho_2)$ and strictly positive over the open interval $(\rho_2, +\infty)$. Thus, the optimal decision depends on the value of c (which in turn is a function of the error probability p):

- If $c \in (1, \rho_2)$, then the Bayesian optimal decision for (T, θ) is Right.
- If $c = \rho_2$, then both Left and Right are Bayesian optimal decisions for (T, θ).
- If $c \in (\rho_2, +\infty)$, then the Bayesian optimal decision for (T, θ) is Left.

■

3.3.2 Arbitrarily many flips as p varies

In section 3.3.1, we gave an example in which the optimal decision flips once as p varies from 0 to 1/2. In this section, we show that the decision may flip an arbitrary number of times. In particular, for each $m > 0$, we construct a tree T for which the optimal decision flips $m - 1$ times as p varies from 0 to 1/2.

The tree T will be built by embedding copies of trees with paths of various numbers of 1's. By adjusting the numbers of copies of each such tree placed in the left and right subtree of T, we will be able to control rather precisely both the degree of $q_T(c)$ and the placement of its roots. The claimed result follows by causing $m - 1$ simple roots to fall in the open interval $(1, +\infty)$.

Let $\mathcal{T} = \{(T_i, \theta_i) : i = 1, 2, \ldots, m\}$ be a collection of labelled trees such that for each i, the root of T_i is labelled 1, and the maximum number of nodes labelled 1 in any path from the root to a leaf in T_i is equal to i. The trees (T_i, θ_i) form the basis of our construction.

Let $\mathbf{s} = (s_1, s_2, \ldots, s_m) \in \mathcal{N}^m$ be an m-dimensional vector of non-negative integers. $\mathcal{T}$ and $\mathbf{s}$ define a forest $\mathcal{F}$ of trees which contains s_i copies of labelled tree (T_i, θ_i) for each i. In the following, we show how, given such a forest $\mathcal{F}$, we can construct a single labelled tree (X, θ_X) satisfying the following:

Path Property: The number of paths in (X, θ_X) with exactly j nodes labelled 1 is equal to the number of paths in the forest $\mathcal{F}$ with exactly j nodes labelled 1, for all $j = 1, 2, \ldots, m$.

The eventual construction of T will then be performed as follows. First, we construct two forests $\mathcal{F}_s$ and $\mathcal{F}_t$ for appropriate m-dimensional vectors $\mathbf{s}$ and $\mathbf{t}$. Next we construct trees (L, θ_L) and (R, θ_R) having the path property with respect to $\mathcal{F}_s$ and $\mathcal{F}_t$, respectively. Finally, we construct T by choosing a root node and making L and R the left and right subtrees, respectively. With appropriate choices of $\mathbf{s}$ and $\mathbf{t}$, the polynomial $q_T(c)$ can be made to coincide with an arbitrary polynomial $q(c)$ with integer coefficients, up to a constant factor. The details follow.

Let X be a complete binary tree, such that the number of leaves in X is greater or equal to the total number of leaves in all trees of the forest $\mathcal{F}$. Let $roots(\mathcal{F})$ be the set of all roots in the forest $\mathcal{F}$. We define an embedding $e: roots(\mathcal{F}) \rightarrow nodes(X)$ which maps each root of a tree in the forest $\mathcal{F}$ to a node of X. This embedding will satisfy the following two properties:

- If r is the root of a tree of height[3] k in the forest $\mathcal{F}$, then its image $e(x)$ has height k in X.
- If $r, r' \in roots(\mathcal{F})$ and $r \neq r'$, then $e(r) \neq e(r')$, and neither is a descendant of the other in X.

It is obvious that e can be naturally extended to an embedding $\hat{e}$ of all nodes of trees in $\mathcal{F}$ into the nodes of X. We define the labelling θ_X as follows: for all $x \in nodes(X)$,

$$\theta_X(x) = \begin{cases} \theta_i(\hat{e}^{-1}(x)) & \text{if } x \in \hat{e}(nodes(T_i)) \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

The labelled tree (X, θ_X) satisfies the desired property by construction.

Let α_{ij} be the number of paths with exactly j nodes labelled 1 in (T_i, θ_i) and let β_j be the total number of paths with exactly j nodes labelled 1 in $\mathcal{F}$. Then,

$$\beta_j = \sum_{i=j}^{m} s_i \alpha_{ij} \tag{32}$$

for all $j = 1, 2, \ldots, m$. Since β_j is also the total number of paths in (X, θ_X) with exactly j nodes labelled 1, $\Phi_X(c)$ can be written in the form:

$$\Phi_X(c) = \sum_{j=0}^{m} \beta_j c^j \tag{33}$$

[3]The *height* of a node in a tree is its maximum distance to a leaf.

where β_0 is the number of leaves in X that are not images of any node x in the forest $\mathcal{F}$ under $\hat{e}$. We note that $\Phi_X(1)$ is equal to the total number of leaves in X.

Consider now a pair of vectors $\mathbf{s} = (s_1, s_2, \ldots, s_m)$, $\mathbf{t} = (t_1, t_2, \ldots, t_m) \in \mathcal{N}^m$. The collection of trees $\mathcal{T}$ of the previous construction and the two vectors $\mathbf{s}$ and $\mathbf{t}$ define two forests, $\mathcal{F}_\mathbf{s}$ and $\mathcal{F}_\mathbf{t}$. We use $\mathcal{F}_\mathbf{s}$ and $\mathcal{F}_\mathbf{t}$ to construct the left subtree (L, θ_L) and right subtree (R, θ_R) of a bigger labelled tree (T, θ). We choose L and R to have the same size, such that the number of leaves in L or R is greater than or equal to the maximum of the total number of leaves in the forests $\mathcal{F}_\mathbf{s}$ and $\mathcal{F}_\mathbf{t}$. For all $x \in nodes(T)$, define:

$$\theta(x) = \begin{cases} \theta_L(x) & \text{if } x \in nodes(L) \\ \theta_R(x) & \text{if } x \in nodes(R) \\ 0 & \text{if } x \text{ is the root of } T \end{cases} \tag{34}$$

We have thus defined a labelled tree (T, θ).

Lemma 13 *Let $\mathbf{s}, \mathbf{t} \in \mathcal{N}^m$, and let $q_T(c) = \sum_{j=0}^{m} \gamma_j c^j$ be the polynomial of the labelled tree (T, θ) constructed as above. Then:*

$$\gamma_j = \sum_{i=j}^{m} (s_i - t_i)\alpha_{ij} \tag{35}$$

for all $j = 1, 2, \ldots, m$, and

$$\gamma_0 = -\sum_{j=1}^{m} \gamma_j \tag{36}$$

Proof: From equations 27, 32, and 33, we have:

$$\begin{aligned} \gamma_j &= \sum_{i=j}^{m} s_i \alpha_{ij} - \sum_{i=j}^{m} t_i \alpha_{ij} \\ &= \sum_{i=j}^{m} (s_i - t_i)\alpha_{ij} \end{aligned}$$

for $j = 1, 2, \ldots, m$, establishing equation 35. $q_T(1)$ is equal to the number of leaves in L minus the number of leaves in R. Since L and R were chosen to have the same size, $q_T(1) = 0$, and equation 36 follows. ■

Theorem 14 *Let $q(c)$ be a polynomial of degree m with integer coefficients, such that $q(1) = 0$. Then there exists a labelled tree (T, θ) and a positive integer d such that $q_T(c) = d \cdot q(c)$ for all c, i.e., $q_T = d \cdot q$ as polynomials. In particular, q_T and q have the same roots.*

Proof: Let $q(c)$ be as in the theorem. Write $q(c) = \sum_{i=0}^{m} \delta_i c^i$, where $\delta_i \in \mathcal{Z}$. We compute a pair of vectors $\mathbf{s}$, $\mathbf{t}$, and a positive integer d so that the labelled tree (T, θ) defined by $\mathbf{s}$ and $\mathbf{t}$ as sketched above has polynomial $q_T(c) = d \cdot q(c)$.

Consider the following system of linear equations:

$$\sum_{i=j}^{m} \alpha_{ij} x_i = \delta_j \tag{37}$$

for all $j = 1, 2, \ldots, m$. This system always has a rational solution $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ because it is upper triangular and all α_{ij} are positive integers. Let d be the least common multiple of the denominators of the x_i's. Then $d \cdot x_i \in \mathcal{Z}$ for all i, and $\mathbf{x}' = (d \cdot x_1, d \cdot x_2, \ldots, d \cdot x_m)$ solves the system of linear equations:

$$\sum_{i=j}^{m} \alpha_{ij} x_i' = d \cdot \delta_j \tag{38}$$

for all $j = 1, 2, \ldots, m$.

Now, define vectors $\mathbf{s}$ and $\mathbf{t}$ as follows. For all $i = 1, 2, \ldots, m$,

- if $x_i \geq 0$, set $s_i = d \cdot x_i$ and $t_i = 0$;
- if $x_i < 0$, set $s_i = 0$ and $t_i = -d \cdot x_i$.

It is obvious that $\mathbf{s}, \mathbf{t} \in \mathcal{N}^m$ and $d \cdot x_i = s_i - t_i$, for all $i = 1, 2, \ldots, m$. Using equations 35 and 38, we have

$$\gamma_j = \sum_{i=j}^{m} \alpha_{ij} (d \cdot x_i) = d \cdot \delta_j$$

for all $j = 1, 2, \ldots, m$. Because both $q_T(c)$ and $q(c)$ have a root at 1, $\gamma_0 = d \cdot \delta_0$. Hence, $q_T(c) = d \cdot q(c)$ as desired. ■

As a result of Theorem 14, we can select a polynomial $q(c)$ of degree m with one root at 1 and $m - 1$ simple roots in the interval $(1, +\infty)$, so that the Bayesian optimal decision for the corresponding labelled tree (T, θ) flips (from Left to Right or from Right to Left) m times as c increases continuously in the interval $(1, +\infty)$. This observation is stated formally in the following corollary of Theorem 14.

Corollary 15 *For any $m \in \mathcal{N}$, there exist a labelled tree (T, θ) and $m - 1$ thresholds $0 < \tau_1 < \tau_2 < \ldots < \tau_{m-1} < 1/2$ defining intervals $I_0 = (0, \tau_1), I_1 = (\tau_1, \tau_2), \ldots, I_{m-1} = (\tau_{m-1}, 1/2)$ such that:*

1. *For all $j = 0, 1, \ldots, m$, the Bayesian optimal decision for (T, θ) is the same for all error probabilities $p \in I_j$.*
2. *For all $j = 1, \ldots, m-1$, if the Bayesian optimal decision for error probability in the interval I_{j-1} is Left (Right), then the Bayesian optimal decision for error probability in the interval I_j is Right (Left).*

Proof: Consider the polynomial $q(c) = (c-1)(c-2)\ldots(c-(m-1))(c-m)$. Obviously, all the coefficients of $q(c)$ are integers, and the m roots of $q(c)$ are the numbers $1, 2, \ldots, m$. By Theorem 14, we can find a tree (T, θ) and a positive integer d such that $q_T(c) = d \cdot q(c)$. Then $q_T(c)$ has also has simple roots $1, 2, \ldots, m$, all but one of which falls in the interval $(1, +\infty)$. Let $\tau_i = p(m + 1 - i)$ for $i = 1, 2, \ldots, m - 1$. Since $p(c)$ is a continuous and monotonic decreasing function, $\tau_1, \tau_2, \ldots, \tau_{m-1}$ satisfy properties (1) and (2) above.

■

4 Game II: Choosing a Leaf

In this section, we consider a variant of Game I where the player is asked to choose a whole path from the root to a leaf of the game tree.

Game II: Let (T, Λ) be a MaxTree with Λ a random labelling following the probability distribution $\mathcal{P}$. Let λ be the outcome of Λ and let θ be a corrupted view of λ with error probability p. Given (T, θ, p), the player is asked to choose a leaf $\alpha \in \mathit{leaves}(T)$. If α is labelled 1 under λ, the player wins; otherwise, she loses.

We fix the underlying probability distribution $\mathcal{P}$ for Λ to be the one defined by equation 1 in section 3, that is, Λ is uniformly distributed among proper labellings in which a single leaf is labelled 1.

4.1 Probabilistic Analysis of Game II

We provide an exact probabilistic analysis of Game II. Given the corrupted view of the game tree and the a priori information about the underlying distribution of labellings, we compute the conditional probability of winning the game for all choices of leaves in T.

Let m be the number of nodes with observed label 1 and n the total number of nodes in (T, θ). Define:

$$b = (1-p)^{n-(m+k+1)} p^{m+k+1}$$

where k is the depth of T. Note that b is a constant which depends only on the value of p and the corrupted labelling θ of the game tree.

Lemma 16 *Let (T, λ) be a MaxTree and let θ be a corrupted view of λ with error probability p. If $\alpha \in \mathit{leaves}(T)$ and π_α is the path from the root of T to the leaf α, then:*

$$\mathbf{prob}[\theta \mid \lambda^\alpha] = b \cdot c^{f(\pi_\alpha)} \tag{39}$$

where $c = ((1-p)/p)^2$.

Proof: Let λ^α be the actual labelling of T. Then, for every $x \in \mathit{nodes}(T)$, the probability that the observed label $\theta(x)$ agrees with the actual label $\lambda^\alpha(x)$ is $(1-p)$, and the probability that the two labels disagree is p. Since the label corruptions happen independently of each other, we can compute the probability of any corrupted view θ by taking the product of the probabilities of the individual label corruptions.

$$\mathbf{prob}[\theta \mid \lambda^\alpha] = \left(\prod_{x \in \pi_\alpha} (1-p)^{\theta(x)} p^{1-\theta(x)} \right) \cdot \left(\prod_{x \in (T - \pi_\alpha)} (1-p)^{1-\theta(x)} p^{\theta(x)} \right) \tag{40}$$

where the first product ranges over all nodes x on the path π_α, and the second product ranges over all nodes x in any other path of T. However, the second product can be rewritten as follows:

$$\prod_{x \in (T - \pi_\alpha)} (1-p)^{1-\theta(x)} p^{\theta(x)} = \frac{\prod_{x \in T} (1-p)^{1-\theta(x)} p^{\theta(x)}}{\prod_{x \in \pi_\alpha} (1-p)^{1-\theta(x)} p^{\theta(x)}} \tag{41}$$

Substituting 41 in equation 40, we obtain:

$$\begin{aligned}
\mathbf{prob}[\theta \mid \lambda^\alpha] &= \left(\prod_{x \in T} (1-p)^{1-\theta(x)} p^{\theta(x)}\right) \cdot \left(\prod_{x \in \pi_\alpha} \left(\frac{1-p}{p}\right)^{\theta(x)} \cdot \left(\frac{p}{1-p}\right)^{1-\theta(x)}\right) \\
&= (1-p)^{n-m} p^m \cdot \left(\frac{p}{1-p}\right)^{(k+1)} \cdot \prod_{x \in \pi_\alpha} \left(\frac{1-p}{p}\right)^{2\theta(x)} \\
&= b \cdot c^{f(\pi_\alpha)} \qquad (42)
\end{aligned}$$

■

Theorem 17 *Let (T, λ) be a MaxTree and let θ be a corrupted view of λ with error probability p. Then, for all $\alpha \in leaves(T)$,*

$$\mathbf{prob}[\lambda^\alpha \mid \theta] = \frac{c^{f(\pi_\alpha)}}{\sum_\beta c^{f(\pi_\beta)}} \qquad (43)$$

where the sum in the denominator ranges over all $\beta \in leaves(T)$.

Proof: Since λ was chosen from the probability distribution $\mathcal{P}$, λ is always winning. Furthermore, it is equally probable that any one of $\beta \in leaves(T)$ is labelled 1 under λ; that is,

$$\mathbf{prob}[\theta] = \frac{1}{2^k} \sum_\beta \mathbf{prob}[\theta \mid \lambda^\beta] \qquad (44)$$

Substituting $\mathbf{prob}[\theta \mid \lambda^\beta]$ from equation 42, we obtain:

$$\mathbf{prob}[\theta] = \frac{b}{2^k} \sum_\beta c^{f(\pi_\beta)} \qquad (45)$$

Finally, we use *Bayes' Theorem* to combine formulas 42 and 45, in order to compute $\mathbf{prob}[\lambda^\alpha \mid \theta]$:

$$\begin{aligned}
\mathbf{prob}[\lambda^\alpha \mid \theta] &= \frac{\mathbf{prob}[\theta \mid \lambda^\alpha] \cdot \mathbf{prob}[\lambda^\alpha]}{\mathbf{prob}[\theta]} \\
&= \frac{c^{f(\pi_\alpha)}}{\sum_\beta c^{f(\pi_\beta)}}
\end{aligned}$$

■

Corollary 18 *Let (T, λ) be a MaxTree and let θ be a corrupted view of λ with error probability p. If $\alpha, \beta \in leaves(T)$, then:*

$$\mathbf{prob}[\lambda^\alpha \mid \theta] \geq \mathbf{prob}[\lambda^\beta \mid \theta] \;\; \textit{iff} \;\; f(\pi_\alpha) \geq f(\pi_\beta) \qquad (46)$$

Corollary 19 *For all instances of Game II, the Bayesian optimal decision does not depend on the value of the error probability p.*

ALGORITHM B_1

Input:

T, a complete binary tree
θ, a corrupted view of T
p, the error probability of the corruption

Output:

α, a leaf of T

Description:

Step 1:

Compute the values of $f(\pi_\beta)$, for all $\beta \in leaves(T)$, as follows: Working down from the root, compute for each node β in T the number of nodes labelled 1 by θ on the path from β to the root. For $\beta \in leaves(T)$, this number is $f(\pi_\beta)$.

Step 2:

Compute $\max\{f(\pi_\beta), \beta \in leaves(T)\}$.
Choose any leaf α, such that $f(\pi_\alpha)$ is maximum.

Figure 5: A Bayesian optimal algorithm for Game II.

4.2 A Bayesian Optimal Algorithm for Game II

We use Corollary 18 to design algorithm B_1, shown in Figure 5, which is Bayesian optimal for Game II. The player computes the number of nodes with observed label 1 on every path from the root of the game tree to a leaf and chooses the path that corresponds to the maximum. Algorithm B_1 is much easier than the Bayesian optimal algorithms A_1 and A_2 for Game I, in that it involves no arithmetic other than counting.

Theorem 20 *Let B_1 be the algorithm shown in Figure 5. Let (T, λ) be a MaxTree of n nodes and let θ be a corrupted view of λ with error probability p. On input (T, θ, p), algorithm B_1 requires $O(n)$ additions and no multiplications to compute the Bayesian optimal decision for Game II.*

Proof: Step 1 of algorithm B_1 is the same as step 2 of algorithm A_2, which was previously shown to require only $O(n)$ additions. ■

4.3 Similar Games—Different Strategies

In section 3.3, we showed that the Bayesian optimal strategy for Game I—choosing a subtree—depends on the error probability p. In other words, in order for the player to compute an optimal decision, it is necessary for her to have some knowledge of the accuracy of the data at hand. On the other hand, in section 4.2, we showed that the optimal strategy for Game II—choosing a leaf—does not depend on p. This observation leads us to an interesting, but somewhat counterintuitive, result: repeated application of a Bayesian optimal decision rule for Game I does not give a Bayesian optimal decision rule for Game II.

Let (T, θ, p) be an instance of Game II where T is of depth k. Consider the following decision algorithm B_2 for playing Game II:

- Set $X = T$;
- Repeat k times: compute the Bayesian optimal decision Y for the instance (X, θ_X, p) of Game I and set $X = Y$;
- Output X.

Theorem 21 *B_2 is not a Bayesian optimal decision algorithm for Game II.*

Proof: Let (T, θ) be the labelled tree of Figure 4 and let $c \in (1, \rho_2)$.

The Bayesian optimal decision for Game II on input (T, θ, p) is the leftmost leaf, since the path from the root of T to the leftmost leaf has the maximum number of nodes with observed label 1 over all such paths.

However, in Theorem 12, we showed that the Bayesian optimal decision for Game I on input (T, θ, p) is Right. As a result, on input (T, θ, p), B_2 chooses a leaf in the right subtree of T. Hence, decision algorithm B_2 is non-optimal. ■

5 Conclusions and Future Work

In this paper, we look at the problem of game-playing/decision-making based on information that is inaccurate because of the presence of random noise. In order for the player in our games to compute a rational decision, she needs to somehow make use of all of all available information. A *Bayesian optimal* strategy accomplishes this since it maximizes the expected value of the player's choice. We show that the Bayesian optimal decision for these games is both easy to compute and expressible by simple, easily-understood formulas which allow for some interesting observations:

- In answering the question "Which subtree has the winning leaf?", the player can make a better decision if she also has some knowledge of the accuracy of the data. Consequently, any algorithm that bases its decision simply on the corrupted view of the game tree will be inferior to one that also makes use of knowledge of the error probabilities.
- In answering the question "Which is the winning leaf?", consistency of the observed data is most important, and the error probability does not affect the optimal decision.

Certain simplifying assumptions in our model seem to make our results very restrictive. For example, we assume that our trees have uniform depth and constant branching factor of two. We also assume a uniform error probability p at each node of the tree and binary node labels in the corrupted view. Geanakoplos and Gray show that these assumptions can be removed and similar results can still be obtained [GG91a, GG91b].

On the other hand, certain other assumptions in our probabilistic setting are difficult to eliminate. For example, the independence of errors among the nodes can be easily characterized as unrealistic, yet removing it entirely makes the problem seem intractable. Perhaps it might still be possible to handle simple patterns of correlation between parent-child nodes or sibling nodes—this is one possible direction for further research. Other

future directions include generalizing these results to two-person games and to iterated (multi-move) games, where new information becomes available after each move.

Finally, we would like to apply Bayesian optimal reasoning to other problems of a similar probabilistic flavor, where redundancy of the data can be used to offset the effect of data corruption. One practical example, pointed out to us by Linda Shapiro, is in computer vision, where noise in the visual data is an everpresent problem in the recognition of the underlying physical objects or patterns.

References

[GG91a] John Geanakoplos and Larry Gray, June 1991. Personal communication.

[GG91b] John Geanakoplos and Larry Gray. When seeing further is not seeing better. Manuscript, July 1991.

[Pea83] Judea Pearl. On the nature of pathology in game searching. *Artificial Intelligence*, 20:427–453, 1983.

[Ros76] Sheldon Ross. *A first course in Probability.* Macmillan, New York, NY, 1976.

[Shu82] Martin Shubik. *Game Theory in the social sciences.* MIT Press, Cambridge, MA, 1982.