Proceedings of Machine Learning Research 83:1–40, 2018 Submitted 02/2018; Published 04/2018

# Learning Decision Trees with Stochastic Linear Classifiers

**Tom Jurgenson** TOM.JURGENSON@GMAIL.COM
*Blavatnik School of Computer Science, Tel Aviv University*

**Yishay Mansour** MANSOUR.YISHAY@GMAIL.COM
*Blavatnik School of Computer Science, Tel Aviv University and Google Research*

**Editor:** Mehryar Mohri and Karthik Sridharan

## Abstract

In this work we propose a top-down decision tree learning algorithm with a class of linear classifiers called stochastic linear classifiers as the internal nodes' hypothesis class. To this end, we derive efficient algorithms for minimizing the Gini index for this class for each internal node, although the problem is non-convex. Moreover, the proposed algorithm has a theoretical guarantee under the weak stochastic hypothesis assumption.

## 1. Introduction

Decision trees have been an influential paradigm from the early days of machine learning. On the one hand, they offer a "humanly understandable" model, and on the other, provide an effective non-convex model. Most decision tree algorithms use a top-down approach to building the decision tree, where each internal node is assigned a hypothesis from some hypothesis class $H_I$. The most notable class is that of *decision stumps*, namely, a threshold over a single attribute, e.g., $x_i \geq \theta$. The decision tree algorithm determines which hypothesis to assign to each internal tree node, to the most part they perform it by minimizing locally a convex function called *splitting criteria*. Decision tree algorithms differ in the splitting criteria they use, for example, the Gini index is used in CART (Breiman et al., 1984) and the binary entropy is used in ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). Minimizing the splitting criteria in a single internal node is a non convex problem. Therefore a significant benefit of decision stumps is that the size of the hypothesis class, given a sample of size $m$, is only $dm$, where $d$ is the number if attributes (for each attribute there are at most $m$ distinct values in $m$ sized sample). Thus, in order to minimize the splitting criteria one could iterate over all decision stumps in linear time.

From a computational perspective, proper learning of decision tree is hard (Alekhnovich et al., 2008). Kearns and Mansour (1999) introduced a framework to analyze popular decision tree algorithms based on the Weak Hypothesis Assumption (see, Schapire (1990)). The Weak Hypothesis Assumption states, that for any distribution, one can find a hypothesis in the class that achieves better than random guessing. The main result of their work shows that decision trees can be used to boost weak learners to strong ones. Qualitatively, assuming the weak learners always have bias at least $\gamma$, decision trees achieve an error of $\epsilon$ with size of $(1/\epsilon)^{O\left((\gamma\epsilon)^{-2}\right)}$, for the Gini index, $(1/\epsilon)^{O\left(\gamma^{-2}\log\left(\epsilon^{-1}\right)\right)}$, for the binary entropy and $(1/\epsilon)^{O\left(\gamma^{-2}\right)}$ (which is polynomial in $1/\epsilon$) for a newly introduced splitting criteria. The framework of Kearns and Mansour (1999) may also encompass

stochastic classifiers, which are used as the hypothesis class for internal decision tree nodes in our work.

Decision tree learning algorithms that assign to each internal node a linear classifier are named *oblique decision trees*, and were originally proposed in CART-LC and developed by Murthy et al. (1994) and Heath et al. (1993). A clear drawback in oblique trees is that the size of the hypothesis class is no longer linear but exponential in the dimension. Therefore the splitting criteria minimization is potentially a hard computational task since enumeration over all possible classifiers is no longer possible in polynomial time, and the various algorithms resort to methods which converge to a local minima. From a complexity point of view, oblique decision trees with three internal nodes encode three-node neural networks, which is computationally hard to learn (Blum and Rivest, 1993).

**Our Contributions:** Our main goal is to have an efficient algorithm for selecting a hypothesis in each internal tree node, namely, being able to efficiently minimize the splitting criteria. We show how to perform this task efficiently for the class of *stochastic linear classifiers*, which is a stochastic version of linear classifiers. We show how to efficiently minimize the Gini index for stochastic linear classifiers, and use our efficient Gini index minimization as part of an overall decision tree learning algorithm.

Our classifier has a few advantages over other decision tree models:

1. Unlike decision trees with decision stumps, stochastic linear classifiers can generate complex decision boundaries.

2. Unlike linear classifiers, it produces a hierarchal structure that allows to fit also data which is not linearly separable.

3. Unlike oblique tree, for a single internal node we can efficiently minimize the splitting criteria of stochastic linear classifiers.

On the downside, since the model becomes more complicated it also becomes less interpretable than either linear classifiers or decision trees which are based on decision stumps.

As part of our analysis we show that the optimal solution can be written as a linear combination of two vectors which can be directly computed from the data. However, simple search for the coefficients of the two vectors might fail, and we need to construct a rather involved discretization, in order to identify the near optimal stochastic linear classifier. We also show that the class of stochastic linear classifiers can essentially simulate deterministic linear classifier, as explained in section 5.

Our work is organized as follows: we start by describing the Decision Tree Model and the top down approach to learning Decision Trees in sections 3 and 4 respectively. In section 5 we present the *Stochastic Linear Classifier*. The main derivation of our algorithms is done in Section 6. Section 7 provides empirical evidence for the performance of decision trees with stochastic linear classifiers. Our concluding remarks appear in section 8.

## 2. Related Work

Two of the most popular decision trees algorithms are C4.5 by Quinlan (1993) and CART by Breiman et al. (1984), which do not use linear classifier but rather decision stumps as the internal node classifier class (due to the efficiency in enumeration of the entire class). A general framework for learning a decision trees based on decision stump with a general splitting criteria was introduced in Nock and Nielsen (2009), and a gradient based method was proposed to solve this general formulation.

The idea of using linear classifiers for the hypotheses class of the internal nodes, date back to CART-LC proposed in Murthy et al. (1994) and Heath et al. (1993). The original proposal of CART-LC suggested to do a simple gradient decent, which deterministically reaches a local minima. One of the successful implementation of oblique decision trees is OC1 by Murthy et al. (1994), which uses a combination of gradient decent and randomization to search for a linear classifier in each internal node. Alternative approaches use simulated annealing by Heath et al. (1993) or evolutionary algorithms by Cantú-Paz and Kamath (2003), to generate a linear classifier in each internal node.

There are other approaches to learning based on decision trees. The work of Bennett and Blue (1998) build a three internal-nodes decision tree using a non-convex optimization which also maximizes the margins of the linear classifiers in the three nodes. The work of Wickramarachchi et al. (2016) uses the eigenvector basis to transform the data, and uses decision stumps in the transformed basis (which are linear classifiers in the original basis). The work of Henry et al. (2007) develops a boosting algorithm $G^2$ which is used recursively to first combine decision stumps into a more complex classifier and next to combine these classifiers again to even more complex classifiers.

## 3. Model

Let $X \subset \mathbb{R}^d$ be the domain and $Y = \{0, 1\}$ be the labels. There is an unknown distribution $D$ over $X \times Y$ and samples are drawn i.i.d. from $D$. Given a hypothesis class $H$ the error of $h \in H$ is $\epsilon(h) = \Pr[h(x) \neq y]$ where the distribution is both over the selection of $(x, y) \sim D$ and any randomization of $h$.

Given two hypothesis classes $H_I$ and $H_L$ mapping $X$ to $\{0, 1\}$ we define a class of decision tree $\mathcal{T}(H_I, H_L)$ as follows. A decision tree classifier $T \in \mathcal{T}(H_I, H_L)$ is tree structure such that each internal node $v$ contains a splitting function $h_v \in H_I$ and each leaf $u$ contains a labeling function $h_u \in H_L$. In order to classify an input $x \in X$ using the decision tree $T$, we start at the root $r$ and evaluate $h_r(x)$. Given that $h_r(x) = b \in \{0, 1\}$ we continue recursively with the subtree rooted at the child $v_b$. When we reach a leaf $l$ we output $h_l(x)$ as the prediction $T(x)$. In case that $H_I$ includes randomized hypotheses we take the expectation over their outcomes. We denote by $\epsilon(T)$ the error of the tree, and the set of leaves in the tree as $Leaves(T)$. We denote the event that input $x \in X$ reaches node $v \in T$ with $Reach(x, v, T)$, and when clear from the context we will use $Reach(x, v)$.

**Splitting criterion** Intuitively, the splitting criterion $G$, is a function that measures how "pure" a node is. This function is used to rank various possible splits of a leaf $v$ using a hypothesis from $H_I$. Formally, $G$ is a *permissible splitting criterion*: if $G : [0, 1] \to [0, 1]$ and has the following three properties:

1. $G$ is symmetric about $0.5$. Thus, $\forall x \in [0, 1] : G(x) = G(1 - x)$.
2. $G$ is normalized: $G(0.5) = 1$ and $G(0) = G(1) = 0$.
3. $G$ is strictly concave.

Three well known such functions are:

1. Gini index: $G(q) = 4q(1 - q)$,
2. The binary Entropy: $G(q) = -q \log(q) - (1 - q) \log(1 - q)$ and
3. sqrt criterion: $G(q) = 2\sqrt{q(1 - q)}$.

One can verify that any permissible splitting criteria $G$ has the property that $G(x) \geq 2\min\{x, 1-x\}$ and so upper bounds twice the classification error.

## 4. Decision Tree Learning Algorithms

The most common decision tree learning algorithms have a top-down approach, which continuously split leaves in a greedy manner using a hypothesis from $H_I$. The decision of which leaf to split and which hypothesis from $H_I$ to use for the split is governed by the splitting criteria $G$. To clarify the role of the splitting criteria $G$ we start with some definitions, and for simplicity we start with a notation that uses a distribution over examples.

Fix a given decision tree $T$, let $v \in T$, denote $\texttt{weight}(v) = \Pr_{(x,y)\sim D}(Reach(x,v))$, and $\texttt{purity}(v) = \Pr_{(x,y)\sim D}(y = 1 | Reach(x,l))$, which is the probability of a positive label conditioned on reaching node $v$. The score of $G$ with respect to $T$ is defined as follows,

$$G(T) = \sum_{l \in Leaves(T)} \texttt{weight}(l) \cdot G(\texttt{purity}(l)). \tag{1}$$

Essentially, this is the expected value of $G(\texttt{purity}(l))$ when a leaf $l$ is sampled using $D$ over $T$. The motivation to minimize $G(T)$ is that $G$ is an upper bound on the error, and therefore $G(T)$ is an upper bound on the error using $T$.

Denote $Split(T, l, h)$ to be the split of leaf $l$ in $T$ using hypothesis $h \in H_I$ which outputs a modified tree $T'$. In $T'$ the leaf $l$ is replaced by an inner node $v$ which has a hypothesis $h$ and two children $v_0$ and $v_1$, where $v_0$ and $v_1$ are leaves. When using the resulting tree, and input $x$ that reaches $v$ then $h(x) = b$ implies that $x$ continues to node $v_b$. Notice that the split is local, as it influences only the inputs that reached leaf $l$ in $T$.

The selection of which leaf and hypothesis to use is based on greedily minimizing $G(T)$. Consider the change in $G(T)$ following $Split(T, l, h)$

$$\begin{aligned} \Delta_{l,h} =& G(T) - G(Split(T,l,h)) \\ =& \texttt{weight}(l) \cdot G(\texttt{purity}(l)) - \sum_{b \in \{0,1\}} \texttt{weight}(v_b) \cdot G(\texttt{purity}(v_b)) \end{aligned} \tag{2}$$

where $v_0$ and $v_1$ are the leaves that result from splitting the leaf $l$. The algorithm would need to find for each leaf $l$ a hypothesis $h_l \in H_I$ which maximizes $\Delta_{l,h}$. This can be done by considering explicitly all the hypotheses in $H_I$, in the case where $H_I$ is small, or by using some optimization oracle. For now, we will abstract away this issue and assume that there is an oracle that given $l$ returns $h_l \in H_I$ which maximizes $\Delta_{l,h}$, hence minimize $G(T)$.

We now describe the Top-Down approach - an iterative process of building a decision tree. Initially, the tree $T$ is comprised of just the root node $r$. Split $r$ using the hypothesis $h_r$ which maximizes $\Delta_{r,h}$ (thus minimizing $G(T)$). In iteration $t$, when we have a current tree $T_t$, for each leaf $l$ of $T_t$ compute $h_l = \arg\max_{h \in H_I} \Delta_{l,h}$. Next, select the leaf with the largest decrease $l_t = \arg\max_l \Delta_{l,h_l}$, and set $T_{t+1} = Split(T, l_t, h_{l_t})$.

Upon termination, the algorithm assigns a hypothesis from $H_L$ to each leaf. Many decision tree learning algorithm simply use constant hypothesis which is a label, i.e., $\{0, 1\}$. Our framework allows for a more elaborate hypothesis at each leaf, we only assume that the error of the hypothesis at leaf $l$ is at most $\min\{x, 1-x\}$ where $x = \texttt{purity}(l)$.

**Using a sample:** Clearly the algorithms do not have access to the true distribution $D$ but only to a sample $S$. Let $S$ be a sample of examples sampled from $D$ ($S$ is a multiset) and let $D_S$ be the empirical distribution induced by $S$. We redefine the previous quantities using $D_S$. Specifically,

$$\texttt{weight}(v) = \Pr_{(x,y)\sim D_S}(Reach(x,v)) = \frac{|\{(x,y) \in S : Reach(x,v) = TRUE\}|}{|S|}$$

$$\texttt{purity}(v) = \Pr_{(x,y)\sim D_S}(y = 1 | Reach(x,l)) = \frac{|\{(x,y) \in S : Reach(x,v) = TRUE, y = 1\}|}{|\{(x,y) \in S : Reach(x,v) = TRUE\}|}$$

The expressions for $G(T)$ and $\Delta_{l,h}$ in equations (1) and (2) are maintained. This allows the use of the Top-Down approach as specified without any further modifications.

**Stochastic Hypothesis:** We extend the setting to allow for a stochastic hypothesis. A major difference is that in the deterministic case the tree partitions the inputs (according to the leaves) while in the stochastic case we have only a modified distribution over the input induced by the leaves. This implies that given a tree $T$, each input $x$ is associated with a distribution over the leaves of $T$. The probability of reaching a node $v$, whose a path in the tree is $\langle (h_1, b_1)...(h_k, b_k)\rangle$ where $h_i \in H_I$ and $b_i \in \{0, 1\}$, is

$$\Pr_{(x,y)\sim D}(Reach(x,v)) = \Pr\left(\forall i \in [1,k] : \ h_i(x) = b_i\right) = \prod_{i=1}^{k} \Pr\left(h_i(x) = b_i | h_j(x) = b_j, j < i\right)$$

For this reason the modified decision tree learning algorithm keeps for each input $x$ a distribution over the current leaves of $T$ (in contrast, in the deterministic case, each input $x$ has a unique leaf). A detailed description of the top-down decision tree learning algorithm `DTL` for a stochastic hypotheses class is provided in Appendix A.

We note that the work of Kearns and Mansour (1999) for bounding the error of Top-Down DT induction trivially extends to our settings of using stochastic classifiers in the internal nodes. In other words, the stochastic classifier presented in the next section can be boosted by a Top-Down DT process to create a stochastic decision tree with an arbitrarily low error. Details regarding the extension for the stochastic settings are provided in Appendix B.

## 5. Stochastic Linear Classifier

In this section we describe the stochastic linear classifier that is used for the hypothesis class $H_I$, which are the hypotheses used in internal nodes of the tree. We also define basic measures and derive properties over them which will be useful later for our mathematical derivations and algorithms.

Let $S = \{\langle x_i, y_i\rangle\}$ be a sample of size $N$ drawn from $D$. Let $D_v$ be the induced distribution over $S$ in node $v$, i.e., $D_v(x_i) \geq 0$ and $\sum_{i=1}^{N} D_v(x_i) = 1$ . We assume that the inputs $x_i$ are normalized, i.e., $\|x_i\|_2 = 1$. We denote the weight of the positive examples according to $D_v$ by $\rho$, i.e., $\rho = \sum_{i=1}^{N} y_i D_v(x_i)$.

The *Stochastic Linear Classifier* is a weight vector $w$, which has a norm at most 1, i.e., $\|w\|_2 \leq 1$. The classification probability is $\Pr[y = 1|x, w] = \frac{w \cdot x + 1}{2}$ , and since both $\|w\| \leq 1$ and $\|x\| = 1$ then $|w \cdot x| \leq 1$ thus $0 \leq \Pr[y = 1|x, w] \leq 1$.

Two important measures that $w$ induces are presented next. The first is $P_w$, the weight of samples classified as positive by $w$, and the second is $Q_w$ the weight of positive labels in the samples classified

by $w$ as positive. Formally,

$$P_w = \sum_{i=1}^{N} D_v(x_i)\frac{w \cdot x_i + 1}{2} \quad \text{and} \quad Q_w = \sum_{i=1}^{N} D_v(x_i)\frac{w \cdot x_i + 1}{2} y_i. \tag{3}$$

Clearly $0 \leq P_w \leq 1$ since $0 \leq \frac{w \cdot x_i + 1}{2} \leq 1$ and $0 \leq Q_w$. We can upper bound $Q_w$ by $P_w$, since $y_i \leq 1$, and by $\rho$, since $\frac{w \cdot x_i + 1}{2} \leq 1$. Finally, we can also lower bound $Q_w$ by $P_w - (1 - \rho)$ since $P_w - Q_w = \sum_{i=1}^{N} D_v(x_i)\frac{w \cdot x_i + 1}{2}(1 - y_i) \leq \sum_{i=1}^{N} D_v(x_i)(1 - y_i) = 1 - \rho$. Therefore,

$$\max\{0, P_w - 1 + \rho\} \leq Q_w \leq \min\{P_w, \rho\} \ . \tag{4}$$

$P_w$ and $Q_w$ could also be written using a vector notation, which sometimes would be more convenient. For this purpose denote the distribution $D_v \in R^N$ as a vector over the sample inputs, $X \in R^{N \times d}$ the examples matrix and $y \in \{0, 1\}^N$ the labels of the examples. Using this notation ($\odot$ is element-by-element multiplication):

$$P_w = \frac{1}{2} D_v^\top X w + \frac{1}{2} \quad \text{and} \quad Q_w = \frac{1}{2} \left(D_v \odot y\right)^\top X w + \frac{\rho}{2} \tag{5}$$

The values of $P_w$ and $Q_w$ play an important role in evaluating candidate classifiers $w$ by plugging them into the splitting criteria, as explained in more details in the next section.

Note that if $w$ is a deterministic linear classifier with margin $\gamma$, then it implies that when we use $w$ as a stochastic linear classifier we "err" with probability $(1 - \gamma)/2$. We can amplify the success probability by having a tree of depth $O(\gamma^{-2} \log \epsilon^{-1})$ and reducing the error probability to $\epsilon$. In appendix H we prove the following theorem.

**Theorem 5.1** *Let $w$ be a $\gamma$ margin feasible linear separator, then there exists a stochastic tree $T$ with feasible linear separators in the internal nodes such that the error of the tree is bounded by $\epsilon$ and the depth of $T$ is $O\left(\ln\left(\epsilon^{-1}\right)\gamma^{-2}\right)$.*

## 6. Approximately Minimizing the Gini Index Efficiently

In this section, we would like to use the generic Top-Down decision tree learning approach in order to greedily minimize the value of the splitting criteria, and thus minimizing indirectly the prediction error. To achieve that we discuss how we select a stochastic linear classifier which maximizes the information gain $\Delta_{l,w}$ for a given leaf $l$.

Given a leaf $l$, the distribution $D_l$ over the samples that reach $l$, and $\rho$ the probability of the positive examples in $D_l$, we need to find a $w$ that maximizes the drop in the splitting criteria, namely

$$\begin{aligned}
\arg\max_w \Delta_{l,w} &= \arg\max_w \texttt{weight}(l)\left(G(\rho) - \left(P_w \cdot G\left(\frac{Q_w}{P_w}\right) + (1 - P_w)\, G\left(\frac{\rho - Q_w}{1 - P_w}\right)\right)\right) \\
&= \arg\min_w \left(P_w \cdot G\left(\frac{Q_w}{P_w}\right) + (1 - P_w)\, G\left(\frac{\rho - Q_w}{1 - P_w}\right)\right)
\end{aligned}$$

This work focuses on the Gini Index splitting criteria, abbreviated to $GI$. Recall that $GI(x) = 4x(1 - x)$, where $x$ is the probability of positive example. The Weighted Gini Index (WGI) is the GI value after a split using $w$:

$$WGI(P_w, Q_w) = P_w GI\left(\frac{Q_w}{P_w}\right) + (1 - P_w) GI\left(\frac{\rho - Q_w}{1 - P_w}\right) = 4\left(\rho - \frac{Q_w^2}{P_w} - \frac{(\rho - Q_w)^2}{1 - P_w}\right) \tag{6}$$

Note that when $P_w = 0$ or $P_w = 1$, all the examples reach the same leaf, and therefore there is no change in the GI, i.e., $WGI(0, Q_w) = WGI(1, Q_w) = GI(\rho)$.

The function $WGI(p, q)$ is concave (rather than convex). Moreover, both $p$ and $q$ are functions of the weight vector $w$, and we show that WGI is not convex in $w$ (see Appendix G). This implies that one cannot simply plug-in the GI and minimize over the weights $w$.

Our first step is to characterize the the structure of the optimal weight vector $w$:

**Theorem 6.1** *For any distribution $D_l$ let $\boldsymbol{a} = D_v^\top X$, $\boldsymbol{b} = (D_v \odot y)^\top X$ and $w_l^* = \arg\min_{\{w:\|w\|\leq 1\}} WGI(P_w, Q_w)$. There exist constants $\alpha$ and $\beta$ such that for $w = \alpha\boldsymbol{a} + \beta\boldsymbol{b}$, we have $\|w\| \leq \|w_l^*\|$ and both $P_{w_l^*} = P_w$ and $Q_{w_l^*} = Q_w$.*

**Proof** Let $P_{w_l^*} = p$ and $Q_{w_l^*} = q$. Let $w'$ be the solution for $\arg\min \|w\|_2^2$ such that $P_w = p$ and $Q_w = q$. Clearly, $\|w'\| \leq \|w_l^*\|$, $P_{w'} = p, Q_{w'} = q$ and therefore $w'$ is also an optimal feasible solution. Lemma E.1 in Appendix E.1 shows that the solution for this optimization problem is:

$$w' = \frac{(2q-\rho)(\boldsymbol{a}\cdot\boldsymbol{b}) - (2p-1)\|\boldsymbol{b}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a}\cdot\boldsymbol{b})^2}\boldsymbol{a} + \frac{(2p-1)(\boldsymbol{a}\cdot\boldsymbol{b}) - (2q-\rho)\|\boldsymbol{a}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a}\cdot\boldsymbol{b})^2}\boldsymbol{b}$$

■

Note that $\boldsymbol{a} = D_v^\top X$ is the weighted feature average of the data and $\boldsymbol{b} = (D_v \odot y)^\top X$ is the weighted feature average of the positive examples. The above characterization suggests that we can search of an approximate optimal solution by using only linear combinations of $\boldsymbol{a}$ and $\boldsymbol{b}$. Our main goal is to find a weight vector $w$ such that $WGI_w - WGI_{w_l^*} \leq \epsilon$. At a high level we perform the search over the possible values of $P_w$ and $Q_w$. Our various algorithms perform the search in different ways, and get different (incomparable) running times. First we define the approximation criteria:

**Definition 6.1** *Let $S$ be a sample, $D_l$ be a distribution over $S$, and $\epsilon$ be an approximation parameter. An algorithm guarantees an $\epsilon$-approximation for $WGI$ using a stochastic linear classifier $w$, such that $\|w\|_2 \leq 1$, if $WGI_w - WGI_{w^*} \leq \epsilon$, where $w^* = \arg\min_{\{w:\|w\|\leq 1\}} WGI_w$.*

The next theorem summarizes the running time of the algorithms described in the following sections

**Theorem 6.2** *Let $S$ be a sample, $D_v$ be a distribution over $S$, and $\epsilon$ be an approximation parameter. Let $\boldsymbol{a} = D_v^\top X$ and $\boldsymbol{b} = (D_v \odot y)^\top X$. Then: For the case where $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent, algorithms* `FixedPQ` *and* `FixedPSearchQ` *guarantee an $\epsilon$-approximation for $WGI$ using a stochastic linear classifier. Algorithm* `FixedPQ` *runs in time $O(Nd) + O\left(\epsilon^{-2}\log\left(\epsilon^{-1}\right)\right)$, and algorithm* `FixedPSearchQ` *runs in time $O(Nd) + O\left(\frac{d}{\epsilon}\right)$. And for the case where $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, i.e., $\boldsymbol{b} = \lambda\boldsymbol{a}$, algorithm* `DependentWGI` *runs in time $O(Nd)$, and achieves the optimal result for WGI using a stochastic linear classifier.*

**Algorithms for the linearly independent case - overview:** Assume that $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. Both algorithms for this case preforms grid search in WGI's domain. The domain includes pairs $(p, q)$, where $p = P_w$ and $q = Q_w$. As we showed before, the domain includes only a subset of $\{(p, q)|0 \leq p \leq 1, \max(0, \rho + p - 1) \leq q \leq \min(\rho, p)\}$). Since the class of stochastic linear classifier also limits the domain (Appendix E.4) we consider the intersection of these two domains. The resulting domain is partitioned by candidate points (partitioned differently for each

algorithm) and the grid search evaluates the candidate with lowest WGI which also matches a feasible $w$ (i.e., $\|w\|_2 \leq 1$). The result is of the form $w = \alpha \boldsymbol{a} + \beta \boldsymbol{b}$, like the characterization of the optimal weight vector, given in Theorem 6.1.

The two algorithms differ in the way they partition the domain into cells and the way they extract candidate points from these cells. However, both use the following theorem (proved in Appendix C)

**Theorem 6.3** *Given data distribution $D_l$ which has a positive label probability of $\rho$ and a parameter $\epsilon \in (0,1)$ it is possible to partition the domain of WGI: $\{(p,q)|0 \leq p \leq 1, \max(0, \rho + p - 1) \leq q \leq \min(\rho, p)\}$ into $O\left(\epsilon^{-2}\right)$ cells, such that the if $(p_1, q_1)$ and $(p_2, q_2)$ belong to the same cell then $|WGI(p_1, q_1) - WGI(p_2, q_2)| \leq \epsilon$.*

Using the cell partition, enables searching for solutions in the domain of the WGI function directly, and thus derive an approximation of the optimal solution $w^*$. The different cell partitions are illustrated in Figures 7 and 6 in Appendix F.

The main challenge is the case of extreme values of $p$ and their influence on the value of $WGI(p,q)$. If we ignore the dependence between the $p$ and $q$ then the derivatives of $WGI(p,q)$ are unbounded when $p \approx 0$ or $p \approx 1$. We overcome this issue by relating the value $p = P_w$ and $q = Q_w$ through their common parameter $w$, and the structure of the cells. This is what enables us to derive the approximation bound.

**Algorithm** `FixedPQ`**:** First, the domain of WGI is partitioned into trapezoid cells, such that the WGI derivatives are monotone in each cell. To achieve that, the entire domain is first partitioned into four sub-domains (details in Appendix C) each with a different trapezoid cell shape as illustrated in Figure 7 in Appendix F. Next, `FixedPQ` iterates over pairs of $(p,q)$ which are the the trapezoid cells' vertices (see candidate generation below). For every candidate $(p,q)$, the $WGI(p,q)$ is evaluated, and the candidate $(p,q)$ pairs are sorted from the lowest $WGI(p,q)$ to the highest. This sorted list is then scanned until a feasible stochastic linear classifier $\|w\| \leq 1$ is found such that $p = P_w$ and $q = Q_w$.

Notice that computing the value of WGI and testing the feasibility of a pair $(p,q)$ does not require computing the classifier $w$ and can be done in $O(1)$ time (since $w = \alpha \boldsymbol{a} + \beta \boldsymbol{b}$, given $\alpha$, $\beta$, $\|\boldsymbol{a}\|$, $\|\boldsymbol{b}\|$ and $(\boldsymbol{a} \cdot \boldsymbol{b})$ the norm of $w$ can be computed directly from those values in $O(1)$ time without computing the weight vector $w$ explicitly). In `FixedPQ` a weight vector $w$ is computed only once (for the first feasible candidate point $(p,q)$ encountered in the scan) and requires $O(d)$ time. The final complexity is $O(Nd) + O\left(\epsilon^{-2} \log\left(\epsilon^{-1}\right)\right)$, where $O(Nd)$ is for computing $\boldsymbol{a}$ and $\boldsymbol{b}$ and the $O\left(\epsilon^{-2} \log\left(\epsilon^{-1}\right)\right)$ is for sorting $O(\epsilon^{-2})$ candidate points. Appendix F.1 provides pseudo-code, a correctness proof and complexity analysis for `FixedPQ`.

*Candidate generation:* A list of $(p,q)$ pairs is created by first selecting $O\left(\epsilon^{-1}\right)$ evenly spaced points that cover the feasible interval for $P_w$ (and also include $p = \rho, 1 - \rho$). The feasible interval is a subsection of $[0,1]$ that is determined by the data (see Appendix E.4). This ensures that the cells generated are contained within a single sub-domain (more details in Appendix C). Next, for each discretized value of $p$ we define $O\left(\epsilon^{-1}\right)$ evenly spaced values of $q$ in the following two intervals: from $\max(0, \rho + p - 1)$ to $\rho p$, and from $\rho p$ to $\min(p, \rho)$. Finally, the pairs are filtered to include only feasible values (as described in Appendix E.4). This process produces $O\left(\epsilon^{-2}\right)$ pairs.

**Algorithm** `FixedPSearchQ`**:** Algorithm `FixedPSearchQ` iterates over values of $p$ which are the endpoints of mutually exclusive slices of trapezoid cells. Each slice, is the set of trapezoid cells which share the same $p$ range as Figure 7 in Appendix F shows. In this case, the endpoints are the

candidates, and for each endpoint at most two weight vector are computed. Non-feasible points are filtered, and the classifier $w$ for the point that has the lowest WGI is returned. Notice that since we do not have the $q$ values in the candidate stage we cannot compute the WGI before computing a weight vector $w$ (or sort by WGI) as we did in `FixedPQ`. Instead we must compute a weight vector $w$ for each candidate and return the best weight vector $w$. The complexity of this algorithm is $O(Nd) + O\left(\frac{d}{\epsilon}\right)$. Appendix F.2 provides pseudo-code, a correctness proof and complexity analysis for `FixedPSearchQ`.

*Candidate generation:* The candidates are $p$ values that are generated by selecting $O\left(\epsilon^{-1}\right)$ evenly spaced points that cover the feasible interval for $P_w$ as in `FixedPQ`. The main difference is that we do not discretize the $q$ values, but maintain a complete slice of all feasible $q$ values for a given range of $p$ values (more details in Appendix C).

**Algorithm for the linearly independent case:**

**Algorithm `DependentWGI`:** Algorithm `DependentWGI` describes the algorithm for the case where the constraints over $P_w$ and $Q_w$ are dependent, i.e., $\boldsymbol{b} = \lambda \boldsymbol{a}$. Appendix D, shows that the optimal weight vector is $w = \pm\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$, both attaining equal values of WGI. In order to compute this weight vector, we compute $\boldsymbol{a}$ and normalize it in $O(Nd)$ time. We also compute the WGI value in $O(1)$ time. The total time complexity is $O(Nd)$. Appendix F.3 provides pseudo-code, a correctness proof and complexity analysis for `DependentWGI`.

**Summary:** We distinguish between the case that $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent or linearly dependent. Algorithm `DependentWGI` finds the optimal solution for the case where $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, while `FixedPQ` and `FixedPSearchQ` are for the linearly independent case, each using a different search method. Algorithm `FixedPQ` iterates over different possible values of $(p, q)$ pairs and its complexity is: $O(Nd) + O(\epsilon^{-2}\log(\epsilon^{-1}))$. Algorithm `FixedPSearchQ` iterates only over a $p$ candidates, and for each computes the optimal $q$ value. The complexity of `FixedPSearchQ` is: $O(Nd) + O\left(\frac{d}{\epsilon}\right)$. The better running time depends on whether $d$ is larger than $O(\epsilon^{-1}\log(\epsilon^{-1}))$ or not.

Finally, if we assume algorithms `DependentWGI`, `FixedPQ` and `FixedPSearchQ` find a weak learner in $H_I$ as described by the Weak Stochastic Hypothesis Assumption B.1 then we are able to boost the results using the *DTL* algorithm using Theorem B.1. Formally the following theorem holds:

**Theorem 6.4** *Let $H_I$ be the feasible stochastic linear classifiers over input space $X$. Let $\gamma \in (0, 0.5]$, and let $f$ be any target function such that $H_I$ $\gamma$-satisfies the Weak Stochastic Hypothesis Assumption with respect to $f$. Let $D$ be any target distribution, and let $T$ be the tree generated by $DTL(D, f, G, t)$ by using algorithms* `DependentWGI`, `FixedPQ` *and* `FixedPSearchQ` *as the local oracles for $t$ iterations. Then for any target error $\epsilon$, the error $\epsilon(T)$ is less than $\epsilon$ provided that $t \geq \left(\frac{1}{\epsilon}\right)^{c/(\gamma^2\epsilon^2\log(1/\epsilon))}$ and $G(q) = 4q(1-q)$.*

## 7. Empirical Evaluation

In this section we demonstrate various evaluations for our proposed classifier. At first, we compare our algorithm to Linear SVM and a decision tree based on decision stumps, namely, CART. The data is highly-dimensional and non-linearly separable, where the boundaries are not axis aligned. All samples are random unit vectors of dimension $d$. The labels correspond to the XOR of the

| Dataset | CART | Linear SVM | Stochastic Tree |
|---|---|---|---|
| Promoters | 0.8 | 0.87 | **0.91** |
| Heart Disease Data Set | 0.7 | 0.73 | **0.77** |
| Statlog (Heart) Data Set | 0.83 | 0.78 | **0.85** |

Figure 1: Performance of three UCI datasets

samples with two hyperplanes: $w_1 = [\mathbf{1}_{d/2}, \mathbf{1}_{d/2}]$ and $w_2 = [-\mathbf{1}_{d/2}, \mathbf{1}_{d/2}]$ thus the label is $y = sign(x \cdot w_1) \oplus sign(x \cdot w_2)$. Both decision trees were allowed to reach depth of 10 and the sample size is $10,000$. We used various dimension sizes from $d = 6$ to $d = 30$, and the results are plotted in Figure 3 . The results were in line with out intuition. Linear SVM achieves the worst performance since the target is not linear. On the other hand, since CART only considers a single attribute in each split, it requires a significant depth to approximate this high-dimensional XOR. Our method achieves good accuracy since it is not limited to a single hyperplane, as is Linear SVM, nor is it limited to axis aligned decisions, as is CART. We do note, that since our method is stochastic, we need to continuously repeat a hyperplane in order to make a split more significant. This explains why, due to the limited depth of the decision tree, our performance deteriorates as the dimension increases. As a final note, our method can achieve the above performance with only $50$ internal nodes, rather than a depth of $10$ (or $1024$ internal nodes) which CART is allowed to use.

Finally, we demonstrate 3 data-sets that were taken from UCI dataset repository (Lichman, 2013) on which our proposed classifier out preforms Linear SVM and CART. The first dataset *Promoters* contains 106 examples with 57 categorical features and two classes. Since our algorithm runs on numeric data we encode each feature to it's one-hot representation for a total of 228 numerical features. The second dataset is the *Heart Disease Data Set* which contains 303 samples with 13 numeric features. In the original data there are 74 features, but it is common to use the above subset of features. The original data contains one negative class to indicate no disease in the patient, and 4 positive classes each corresponding to a certain disease. We reduced the problem to a positive vs. negative classification, by combining all the diseases to one class. The third dataset *Statlog (Heart) Data Set*, has the same format as the previous only the target variable is already in binary format. There are 13 features and 270 samples in this dataset. The following table in Figure 1 shows the accuracy of all three classifiers used in our experiments, and demonstrates the advantage of our proposed method.

We note that we gave the same depth restriction to CART and our Stochastic Decision Tree, however, our method seems sometimes to require a smaller depth to converge. For the *Promoters* data-set, while CART was still improving at depth $10$, the stochastic tree needed only a depth of $3$ to converge. This along with the comparatively strong result of the Linear SVM may hint that the true decision boundary can be approximated well with a hyperplane which is not axis aligned. For the other two data-sets, *Heart Disease Data Set* and *Statlog (Heart) Data Set* both methods converged at the same depth of $3$ and $5$ correspondingly.

Finally, we refer the reader to Appendix I for empirical results which demonstrate the ability of our proposed classifier to learn a variety of concept classes.

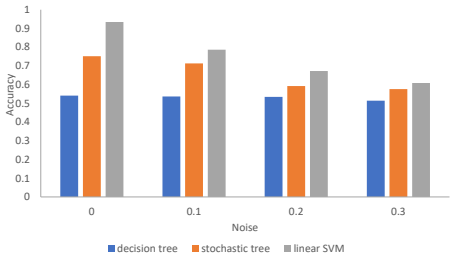


Figure 2: Hyperplane with noise.

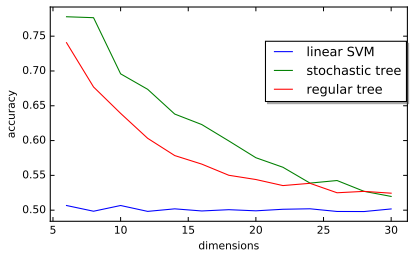


Figure 3: Multi-dimensional XOR

## 8. Conclusion

In this work we introduced a stochastic decision tree learning algorithm based on *stochastic linear classifiers*. The main advantage of our approach is that we can efficiently minimize the Gini index in each internal node of the tree.

Theoretically, the algorithms presented in this work could be applied to different splitting criteria $G$, by finding a cell partition of $P_w$ and $Q_w$ which holds: (1) monotonicity of $G$ inside the cell, and (2) bounded $G$ difference for points inside the cell. Such application, or finding a common framework that is able to partition all splitting criteria, could be an interesting future work.

A different possible follow-up direction would be to explore the benefits of the stochastic linear classifier in other boosting frameworks. Finally, we could consider changing the linear modeling for $\Pr(y|x)$ to a more complex representation, such as a log linear model. This gives rise to an immediate computational challenge of efficiently minimizing the splitting criteria.

## References

Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1):16–34, 2008.

Kristin P Bennett and Jennifer A Blue. A support vector machine approach to decision trees. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 3, pages 2396–2401. IEEE, 1998.

Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is NP-Complete. In *Machine Learning: From Theory to Applications - Cooperative Research at Siemens and MIT*, pages 9–28, 1993.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Erick Cantú-Paz and Chandrika Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 7(1):54–68, 2003.

David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993.

C. Henry, R. Nock, and F. Nielsen. Real boosting *a la Carte* with an application to boosting Oblique Decision Trees. In *IJCAI07*, pages 842–847, 2007.

Michael J. Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. *J. Comput. Syst. Sci.*, 58(1):109–128, 1999.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 1994.

R. Nock and F. Nielsen. Bregman divergences and surrogates for learning. *TPAMI*, 31:2048–2059, 2009.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

J Ross Quinlan. C4. 5: Programming for machine learning. *Morgan Kauffmann*, page 38, 1993.

Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

D.C. Wickramarachchi, B.L. Robertson, M. Reale, C.J. Price, and J. Brown. Hhcart. *Comput. Stat. Data Anal.*, 96(C):12–23, April 2016. ISSN 0167-9473.

## Appendix A. Decision tree learning algorithm: pseudo code

The pseudo code of Algorithm A encapsulates most existing decision tree learning (DTL) algorithms, including ones such as CART by Breiman et al. (1984) and C4.5 by Quinlan (1993). Our exposition of the decision tree algorithms tries to abstract the specific splitting function $G$ and the way to minimize it. We also generalize the standard framework to handle stochastic classifiers. We start with describing the oracles we assume.

The first oracle minimizes the splitting criteria for a hypothesis in an internal node. The function $FitH_I(S, D, G)$ receives as input a sample $S$, a distribution $D$ over the sample $S$ and a splitting criteria $G$. It returns $(h^*, g^*)$ where $h^* \in H_I$ is the hypothesis that minimizes the splitting criteria $G$ over the distribution $D$ on $S$ and $g^*$ is its reduction in the value of the splitting criteria $G$.

The second oracle $FitH_L(S, D)$ receives as input a sample $S$ and a distribution $D$ over the sample $S$ and returns a hypothesis $h \in H_L$ which minimizes the error (and not the splitting criteria $G$).

Third, the function $SplitLeaf(T, l, h)$ receives a decision tree $T$, a leaf $l \in Leaves(T)$ and an hypothesis $h \in H_I$ and returns a new tree where we split leaf $l$ using hypothesis $h$.

The decision tree learning algorithm runs for $t$ iterations. In each iteration, for each leaf $l \in Leaves(T)$ the function $FitH_I$ evaluates the optimal predictor $h_l^*$ for the leaf $l$ along with $g_l^*$, the local reduction in $G$ induced by $h_l^*$. The local reduction is weighted by $w_l$ to obtain the global reduction in $G$, i.e., $w_l g_l^*$, and we select the leaf that minimizes $G$ globally. We update the tree by splitting this leaf, and for the new leaves update the probabilities of inputs to reach each of them. Once $t$ iterations are done and the leaves have been determined, $FitH_L$ selects for each leaf a predictor from $H_L$ by minimizing the error (and not the splitting criteria $G$).

One can observe that our framework encapsulates common decision tree methodologies algorithms, such as C4.5 and CART, as follows: (1) $H_I$ is the family of all decision stumps and $H_L = \{0, 1\}$. (2) The function $FitH_I$ is implemented by evaluating all the decision stumps and selecting the best according to $G$. (3) The function $FitH_L$ sets $y = 0$ or $y = 1$ according to the majority class in the leaf, minimizing the error given a constant label predictor.

We presented a general basic scheme of the *DTL* algorithm. We did not describe other hyperparameters (such as minimal split size) or methods to refine the model (such as pruning) which could be applied to the generic algorithm in order to improve results. Our goal is to focus of the core properties of the DTL which we believe are well represented in our framework.

## Appendix B. Extending the boosting framework to the stochastic settings

In this section we extend the results of Kearns and Mansour (1999) to include the boosting ability of top down algorithms using stochastic classifiers in the internal nodes. Namely, under the assumption

**Assumption B.1 (Weak Stochastic Hypothesis Assumption)** *Let $f$ be any boolean function over the input space $X$. Let $H$ be a class of stochastic boolean functions over $X$. Let $\gamma \in (0, 0.5]$. We say $H$ $\gamma$-satisfies the Weak Stochastic Hypothesis Assumption with respect to $f$ if for every distribution $\bar{D}$ over $X$, there exists an hypothesis $h \in H$ such that $\Pr_{x \sim \bar{D}}(f(x) \neq h(x)) \leq 0.5 - \gamma$ where the probability is both over the distribution $\bar{D}$, and the randomness of $h$.*

We prove the following theorem,

**Algorithm** DTL ($S$ *- sample,* $G$ *splitting criteria,* $t$ *number of iterations*)

Initialize $T$ to be a single-leaf tree with node $r$, for $x \in S$: $p_r(x) = \frac{1}{|S|}$, $w_r = 1$, and $q_r = \frac{|\{x \in S : f(x) = 1\}|}{|S|}$.
**while** $|Leaves(T)| < t$ **do**
  $\Delta_{best} \leftarrow 0$; $best_l \leftarrow \perp$; $best_h \leftarrow \perp$
  **for** *each* $l \in Leaves(T)$*:* **do**
    $h_l^*, g_l^* \leftarrow FitH_I(S, \frac{1}{w_l} p_l(x), G)$
    $\Delta_l \leftarrow w_l G(q_l) - w_l \cdot g_l^*$
    **if** $\Delta_l \geq \Delta_{best}$ **then**
      $\Delta_{best} \leftarrow \Delta_l$; $best_l \leftarrow l$; $best_h \leftarrow h_l^*$
    **end**
  **end**
  **for** $x \in S$ **do**
    $p_{l_1}(x) \leftarrow p_l(x) \cdot \Pr(best_h(x))$
    $p_{l_0}(x) \leftarrow p_l(x) \cdot (1 - \Pr(best_h(x)))$
  **end**
  $w_{l_1} \leftarrow \sum_{x \in S} p_{l_1}(x)$; $w_{l_0} \leftarrow \sum_{x \in S} p_{l_0}(x)$
  $q_{l_1} \leftarrow \sum_{x \in S : f(x) = 1} p_{l_1}(x)$; $q_{l_0} \leftarrow \sum_{x \in S : f(x) = 1} p_{l_0}(x)$
  $T \leftarrow SplitLeaf(T, best_l, best_h)$
**end**
**for** *each* $l \in leaves(T)$*:* **do**
  $h_l \leftarrow FitH_L(S, \frac{1}{w_l} p_l(x))$
**end**
**return** $T$

**Algorithm 1:** Decision tree learning algorithm

**Theorem B.1** *Let $H$ be any class of stochastic boolean functions over input space $X$. Let $\gamma \in (0, 0.5]$, and let $f$ be any target function such that $H$ $\gamma$-satisfies the Weak Stochastic Hypothesis Assumption with respect to $f$. Let $D$ be any target distribution, and let $T$ be the tree generated by $DTL(D, f, G, t)$ with $t$ iterations. Then for any target error $\epsilon$, the error $\epsilon(T)$ is less than $\epsilon$ provided that $G$ is the Gini index and $t \geq \left(\frac{1}{\epsilon}\right)^{c/(\gamma^2\epsilon^2 \log(1/\epsilon))}$*

The above theorem establishes a tradeoff between the desired error $\epsilon$, the bias parameter $\gamma$ of the Weak Stochastic Hypothesis Assumption, the splitting criteria $G$ and the decision tree size $t$.

The proof for B.1 is based on the deterministic case proof by Kearns and Mansour (1999), and we explicitly supply the proof where our work diverges, namely Lemma 2 of Kearns and Mansour (1999) which we substitute by Lemma B.1.

We start with a few notations: Let $f$ be the target function, let $T$ be a decision tree, let $l \in Leaves(T)$ be a leaf in $T$, and let $h \in H_I$. Denote the leaves introduced by $Split(T, l, h)$ by $l_1$ and $l_0$. Recall that $\texttt{purity}(l)$ is the fraction of positive examples that reach leaf $l$ from all of the examples that reach it. We use the following shorthand notations for reoccurring expressions:

$$q = \texttt{purity}(l), p = \texttt{purity}(l_0), r = \texttt{purity}(l_1)$$

$$\tau = \Pr_{(x,y)\sim D}(Reach(x, l_1)|Reach(x, l)) = \Pr_{(x,y)\sim D}(h(x) = 1|Reach(x, l)) = \frac{\texttt{weight}(l_1)}{\texttt{weight}(l)}$$

Since $q = (1 - \tau)p + \tau r$ and $\tau \in [0, 1]$ than without loss of generality we have: $p \leq q \leq r$. Let:

$$\delta = r - p \ .$$

Let $D_l$ be the distribution over input at leaf $l$. We also define $D'_l$ the balanced distribution over inputs in leaf $l$ in which the probability positive (or negative) weights is $1/2$. Formally,

$$D_l(x) = \frac{Reach(x, l)}{\texttt{weight}(l)} D(x)$$

$$D'_l(x) = \begin{cases} \frac{D_l(x)}{2q} & \text{if } f(x) = 1 \\ \frac{D_l(x)}{2(1-q)} & \text{if } f(x) = 0 \end{cases}$$

Recall that the information gain is $\Delta = G(q) - (1 - \tau)G(p) - \tau G(r)$. Note that if either $\delta$ is small or $\tau$ is near 0 or 1 then the information gain is small. Lemma B.1 shows that if $h \in H_I$ is used to split at $l$, and $h$ satisfies the Weak Stochastic Hypothesis Assumption for $D'_l$, then both $\delta$ cannot be too small, and $\tau$ cannot be too close to either 0 or 1.

**Lemma B.1** *Let $p$, $\tau$ and $\delta$ be as defined above for the split $h \in H_I$ at leaf $l$. Let $D_l$ and $D'_l$ also be defined as above for $l$. If the function $h$ satisfies $\Pr_{x\sim D'_l}(h(x) \neq f(x)) \leq 0.5 - \gamma$, Then:*

$$\tau(1 - \tau)\delta \geq 2\gamma q(1 - q) \ .$$

**Proof** We calculate the following expressions in terms of $\tau, r$ and $q$:

$$\Pr_{x\sim D_l}(f(x) = 1, h(x) = 1) = \tau r$$

Therefore,

$$\Pr_{x\sim D_l}(f(x)=0,h(x)=1)=\tau(1-r) \quad \text{and} \quad \Pr_{x\sim D_l}(f(x)=1,h(x)=0)=(1-\tau)p=q-\tau r$$

Next we consider the error terms under the distribution $D'_l$. When changing from the distribution $D_l$ to $D'_l$ we need to scale the expression according to the labels:

$$\Pr_{x\sim D'_l}(f(x)=0,h(x)=1)=\frac{\tau(1-r)}{2(1-q)} \quad \text{and} \quad \Pr_{x\sim D'_l}(f(x)=1,h(x)=0)=\frac{q-r\tau}{2q}$$

Finally the error terms are combined to achieve the total error:

$$\begin{aligned}\Pr_{x\sim D'_l}(f(x)\neq h(x)) &= \Pr_{x\sim D'_l}(f(x)=0,h(x)=1)+\Pr_{x\sim D'_l}(f(x)=1,h(x)=0)\\ &=\frac{\tau(1-r)}{2(1-q)}+\frac{q-r\tau}{2q}=\frac{1}{2}+\frac{\tau}{2}\left(\frac{1-r}{1-q}-\frac{r}{q}\right)=\frac{1}{2}+\frac{\tau}{2}\cdot\frac{q-r}{q(1-q)}\end{aligned}$$

By our assumption on $h$ we have,

$$\Pr_{x\sim D'_l}(f(x)\neq h(x))=\frac{1}{2}+\frac{\tau}{2}\cdot\frac{q-r}{q(1-q)}\leq\frac{1}{2}-\gamma$$

which implies that

$$\frac{\tau}{2}\cdot\frac{r-q}{q(1-q)}\geq\gamma$$

Substituting $r=q+(1-\tau)\delta$ we get the required inequality. ■

We note that, we have in fact provided a proof for 2 other splitting criteria (as was shown in Kearns and Mansour (1999)). Therefore Theorem B.1 also holds for the following conditions on $t$ and $G$:

$$\begin{aligned} t &\geq \left(\frac{1}{\epsilon}\right)^{c/(\gamma^2\epsilon^2\log(1/\epsilon))} && \text{if } G(q)=4q(1-q)\\ t &\geq \left(\frac{1}{\epsilon}\right)^{c\log(1/\epsilon)/\gamma^2} && \text{if } G(q)=H(q)\\ t &\geq \left(\frac{1}{\epsilon}\right)^{c/\gamma^2} && \text{if } G(q)=2\sqrt{q(1-q)}\end{aligned}$$

## Appendix C. Bounding the WGI by regions

In this appendix, we show how to partition the domain of WGI into cells such that the difference between values of the WGI of points in the same cell is bounded by $\epsilon$. Namely, we prove the following theorem.

**Theorem 6.3** *Given data distribution $D_l$ which has a positive label probability of $\rho$ and a parameter $\epsilon\in(0,1)$ it is possible to partition the domain of WGI: $\{(p,q)|0\leq p\leq 1,\max(0,\rho+p-1)\leq q\leq\min(\rho,p)\}$ into $O\left(\epsilon^{-2}\right)$ cells, such that the if $(p_1,q_1)$ and $(p_2,q_2)$ belong to the same cell then $|WGI(p_1,q_1)-WGI(p_2,q_2)|\leq\epsilon$.*

### C.1. Partitioning the WGI to monotone regions

Consider the domain of $WGI$ which is

$$R = \{(p,q)|p \in (0,1), q \in [\max(0, p+\rho-1), \min(\rho, p)]\} \ .$$

Note that the values of $p = 0$ and $p = 1$ are not included. We discuss and add them in Section C.7. We partition $R$ to two sub-domains according to the line $q = \rho p$, namely,

$$R_a = \{(p,q)|p \in (0,1), q \in [\rho p, \min(\rho, p)]\}$$

and

$$R_b = \{(p,q)|p \in (0,1), q \in [\max(0, p+\rho-1), \rho p]\} \ ,$$

as illustrated in Figure 4.

We show that the function WGI is monotone in each of those domains.

**Lemma C.1** *Given a data distribution $D_l$ which has a positive class weight of $\rho$, the WGI function:*

$$WGI(p,q) = 4\left(\rho - \frac{q^2}{p} - \frac{(\rho-q)^2}{1-p}\right)$$

*is monotone in both $p$ and $q$ in both $R_a$ and $R_b$. Specifically, in $R_a$ it is increasing in $p$ and decreasing in $q$ and in $R_b$ it is decreasing in $p$ and increasing in $q$.*

**Proof** Since we are interested only in monotonicity, we can simply consider the function $g(p,q) = -\frac{q^2}{p} - \frac{(\rho-q)^2}{1-p}$. In order to show that $g(p,q)$ is monotone, we first consider where its derivatives vanishes. The derivative with respect to $q$ is

$$\frac{\partial g(p,q)}{\partial q} = -\frac{2q}{p} + \frac{2(\rho-q)}{(1-p)} = \frac{2(\rho p - q)}{p(1-p)}$$

This implies that the derivative vanishes at $q = \rho p$ which is the boundary of the sub-domains $R_a$ and $R_b$, and hence $g(p,q)$ is monotone in $q$ in each of the domains. Also, in $R_a$, since $q > p\rho$, the derivative is negative and in $R_b$, since $q \leq p\rho$, it is positive.

Next we consider the derivative with respect to $p$,

$$\frac{\partial g(p,q)}{\partial p} = \frac{q^2}{p^2} - \frac{(\rho-q)^2}{(1-p)^2} = \frac{(q-\rho p)(q(1-2p)+\rho p)}{p^2(1-p)^2}$$

This implies that the derivative vanishes at $q = \rho p$ and $q = \frac{\rho p}{2p-1}$. The line $q = \rho p$ is a boundary of our regions and so we consider only the other curve.

For $p \in (0, 0.5)$ we have $1 - 2p > 0$. This implies that $q(1-2p) + \rho p > 0$. Since $q > \rho p$ in $R_a$ it implies that $g(p,q)$ is increasing there, and similarly, since $q \leq p\rho$, it is decreasing in $R_b$.

For $p \in (0.5, 1)$ we have $-1 < 1-2p < 0$. For $R_b$, since $q \leq \rho p$ we have that $q(1-2p)+\rho p > 0$ and hence $g(p,q)$ is decreasing in $p$. For $R_a$, we have $q \leq p+\rho-1$. This implies that $q(1-2p)+\rho p \leq (1-p)(2p-1+\rho)$ which is non-negative since $p \in (0.5, 1)$. Therefore $q(p,q)$ is increasing in $R_a$.

Finally, for $p = 0.5$ we have that $\frac{\partial g(p,q)}{\partial p} = \frac{(q-\rho/2)(\rho/2)}{1/16}$ which is positive in $R_a$ and negative in $R_b$ as required. ■

Figure 4 shows the sub-domains $R_a$ and $R_b$.

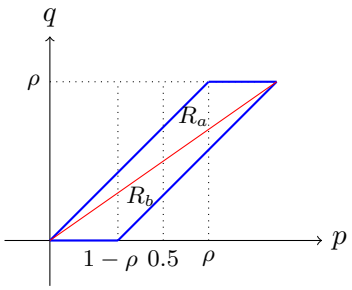


Figure 4: The domain $R$ and the sub-domains $R_a$ and $R_b$ (for $\rho \geq 0.5$)

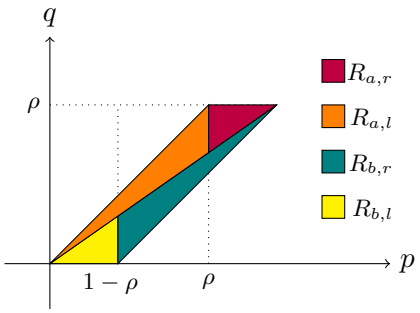


Figure 5: WGI: partitioned into regions for $\rho \geq 0.5$

### C.2. Overview of the partition into cells

We partitioned the domain $R$ of WGI into two sub-domains $R_a$ and $R_b$. According to Theorem C.1 in each of the sub-domains the WGI is monotone in $p$ and $q$. We will further partition each sub-domain in to two parts, and then split them in to cells. We first consider the difference in the value of WGI between two points:

$$\begin{aligned} |WGI(p_1,q_1) - WGI(p_2,q_2)| &= 4\left|\frac{q_2^2}{p_2} + \frac{(\rho-q_2)^2}{1-p_2} - \frac{q_1^2}{p_1} - \frac{(\rho-q_1)^2}{1-p_1}\right| \\ &\leq 4\left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| + 4\left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right| = 4\Delta_1 + 4\Delta_2 \quad (7) \end{aligned}$$

where $\Delta_1 = \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right|$ and $\Delta_2 = \left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right|$.

We are going to consider the case where $\rho \geq 0.5$. The symmetric case of $\rho < 0.5$ can be reduced to this case by switching the labels and deriving the same bound. For the definition of the cells we use the parameters $\epsilon \in (0,1)$ which controls the cell's size.

We next present an analysis for each of four sub-domains, show that in each both $\Delta_1 = O(\epsilon)$ and $\Delta_2 = O(\epsilon)$. In addition each sub-domain will be partitioned in to $O(\epsilon^{-2})$ cells.

### C.3. First region: $R_{b,l}$

Let $R_{b,l} = \{(p,q)|0 < p \leq 1-\rho, 0 \leq q \leq \rho p\}$. Clearly $R_{b,l} \subseteq R_b$, and by Lemma C.1 we know that WGI is decreasing in $p$ and increasing in $q$ in $R_{b,l}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in (0, 1-\rho-\epsilon]$ and $\alpha \in [\epsilon, 1]$. Let

$$cell_{R_{b,l}}(p_c, \alpha) = \{(p,q) | p_c \le p \le p_c + \epsilon, (\alpha-\epsilon)\rho p \le q \le \alpha\rho p\}\,.$$

The cells would have $p_c = i\epsilon$, for $0 \le i \le \frac{1-\rho}{\epsilon} - 1$ and $\alpha = j\epsilon$, for $1 \le j \le 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{b,l}$. Since in $R_{b,l}$ we have $\frac{\partial WGI}{\partial q} \ge 0$, there are $p_1, p_2 \in (0, 1-\rho]$, such that the maximum WGI value in $cell_{R_{b,l}}(p_c, \alpha)$ is obtained at a point $(p_1, q_1) = (p_1, \alpha\rho p_1)$ and the minimum value at $(p_2, q_2) = (p_2, (\alpha-\epsilon)\rho p_2)$.

We bound $\Delta_1$ as follows:

$$\Delta_1 = \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| = \left|\frac{(\alpha-\epsilon)^2\rho^2 p_2^2}{p_2} - \frac{\alpha^2\rho^2 p_1^2}{p_1}\right| \le \rho^2(\alpha^2 |p_2 - p_1| + \epsilon p_2) \le 2\epsilon \tag{8}$$

and $\Delta_2$ as follows,

$$\begin{aligned}
\Delta_2 &= \left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right| = \left|\frac{(\rho-(\alpha-\epsilon)\rho p_2)^2}{1-p_2} - \frac{(\rho-\alpha\rho p_1)^2}{1-p_1}\right| = \rho^2\left|\frac{(1-(\alpha-\epsilon)p_2)^2}{1-p_2} - \frac{(1-\alpha p_1)^2}{1-p_1}\right| \\
&\le \rho^2\left(\overbrace{\alpha^2}^{\le 1}\overbrace{|p_1-p_2|}^{\le \epsilon} + \epsilon\overbrace{|\epsilon-2\alpha|}^{\le 2}\overbrace{|1-p_2|}^{\le 1} + 2\epsilon\overbrace{|2\alpha-1-\epsilon|}^{\le 2} + \frac{\epsilon\overbrace{|2(1-\alpha)+\epsilon|}^{\le 2}}{1-p_2} + \left|\frac{(1-\alpha)^2}{1-p_2} - \frac{(1-\alpha)^2}{1-p_1}\right|\right) \\
&\le \rho^2\left(7\epsilon + \frac{2\epsilon}{1-p_2} + \overbrace{(1-\alpha)^2}^{\le 1}\frac{\overbrace{|p_2-p_1|}^{\le \epsilon}}{(1-p_1)(1-p_2)}\right) \le 7\rho^2\epsilon + \frac{\rho^2\epsilon}{\rho^2} + 2\rho\frac{\rho\epsilon}{\rho} \le 10\epsilon
\end{aligned} \tag{9}$$

where we used the fact that $\rho \le 1$ and $p \le 1-\rho$.

Combining the bounds in (7), (8) and (9) the error for in each cell in $R_{b,l}$ is bounded by $4(2\epsilon + 10\epsilon) = 48\epsilon$.

### C.4. Second region: $R_{b,r}$

Let $R_{b,r} = \{(p,q) | 1-\rho \le p < 1, p+\rho-1 \le q \le \rho p\}$. Clearly $R_{b,r} \subseteq R_b$, and by Lemma C.1 we know that WGI is decreasing in $p$ and increasing in $q$ in $R_{b,r}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in [1-\rho, 1-\epsilon)$ and $\alpha \in [\epsilon, 1]$. Let

$$\begin{aligned}
cell_{R_{b,r}}(p_c, \alpha) = \{&(p,q) : p_c \le p \le p_c + \epsilon, \\
&(\alpha-\epsilon)\rho p + (1-\alpha+\epsilon)(p+\rho-1) \le q \le \alpha\rho p + (1-\alpha)(p+\rho-1)\}\,.
\end{aligned}$$

The cells would have $p_c = i\epsilon$, for $(1-\rho)/\epsilon \le i \le 1/\epsilon - 1$ and $\alpha = j\epsilon$, for $1 \le j \le 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{b,r}$. Since in $R_{b,r}$ we have $\frac{\partial WGI}{\partial q} \ge 0$, there are $p_1, p_2 \in [1-\rho, 1)$, such that the maximum WGI value in $cell_{R_{b,r}}(p_c, \alpha)$ is obtained at a point $(p_1, q_1) = (p_1, \alpha\rho p_1 + (1-\alpha)(p_1+\rho-1))$ and the minimum value at $(p_2, q_2) = (p_2, (\alpha-\epsilon)\rho p_2 + (1-\alpha+\epsilon)(p_2+\rho-1))$

We bound $\Delta_1$ as follows:

$$\begin{aligned}\Delta_1 =& \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| = \left|\frac{((\alpha-\epsilon)\rho p_2 + (1-\alpha+\epsilon)(p_2+\rho-1))^2}{p_2} - \frac{(\alpha\rho p_1 + (1-\alpha)(p_1+\rho-1))^2}{p_1}\right| \\ \leq& \alpha^2\rho^2\left|p_2-p_1\right| + \epsilon\left|\epsilon-2\alpha\right|\rho^2 p_2 + 2\alpha(1-\alpha)\rho\left|p_2-p_1\right| + 2\epsilon\left|2\alpha-\epsilon-1\right|\rho\left|p_2+\rho-1\right| + \\ & (1-\alpha)^2\left|\frac{(p_2+\rho-1)^2}{p_2} - \frac{(p_1+\rho-1)^2}{p_1}\right| + \frac{\epsilon\left|2-2\alpha+\epsilon\right|(p_2+\rho-1)^2}{p_2} \\ \leq& 5.5\epsilon + \left|\frac{(p_2+\rho-1)^2}{p_2} - \frac{(p_1+\rho-1)^2}{p_1}\right| + \frac{2\epsilon(p_2+\rho-1)^2}{p_2} \leq 12.5\epsilon \end{aligned} \tag{10}$$

For $\Delta_2$ we have

$$\begin{aligned}\Delta_2 =& \left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right| \\ =& \left|\frac{(\rho-(\alpha-\epsilon)\rho p_2 - (1-\alpha+\epsilon)(p_2+\rho-1))^2}{1-p_2} - \frac{(\rho-\alpha\rho p_1 - (1-\alpha)(p_1+\rho-1))^2}{1-p_1}\right| \leq 8.5\epsilon \end{aligned} \tag{11}$$

Combining the bounds in (7), (10) and (11) the error for in each cell in $R_{b,l}$ is bounded by $4(12.5\epsilon + 8.5\epsilon) = 84\epsilon$.

### C.5. Third region $R_{a,l}$

Let $R_{a,l} = \{(p,q)|0 < p \leq \rho, \rho p \leq q \leq p\}$. Clearly $R_{a,l} \subseteq R_a$, and by Lemma C.1 we know that WGI is increasing in $p$ and decreasing in $q$ in $R_{a,l}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in (0, \rho-\epsilon]$ and $\alpha \in [\epsilon, 1]$. Let

$$cell_{R_{a,l}}(p_c,\alpha) = \{(p,q)|p_c \leq p \leq p_c+\epsilon, (\alpha-\epsilon)p + (1-\alpha+\epsilon)\rho p \leq q \leq \alpha p + (1-\alpha)\rho p\} \ .$$

The cells would have $p_c = i\epsilon$, for $0 \leq i \leq \rho/\epsilon - 1$ and $\alpha = j\epsilon$, for $1 \leq j \leq 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{a,l}$. Since in $R_{a,l}$ we have $\frac{\partial WGI}{\partial q} \leq 0$, there are $p_1, p_2 \in (0,\rho]$, such that the maximum WGI value in $cell_{R_{a,l}}(p_c,\alpha)$ is obtained at a point $(p_1,q_1) = (p_1, (\alpha-\epsilon)p_1 + (1-\alpha+\epsilon)\rho p_1)$ and the minimum value at $(p_2,q_2) = (p_2, \alpha p_2 + (1-\alpha)\rho p_2)$

We bound $\Delta_1$ as follows,

$$\begin{aligned}\Delta_1 =& \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| = \left|\frac{(\alpha p_2 + (1-\alpha)\rho p_2)^2}{p_2} - \frac{((\alpha-\epsilon)p_1 + (1-\alpha+\epsilon)\rho p_1)^2}{p_1}\right| \\ \leq& \left|p_2-p_1\right|\left|\rho+(1-\rho)\alpha\right|^2 + 2\epsilon p_1\left|\rho+(1-\rho)\alpha\right|\left|\rho-1\right| + \epsilon^2 p_1(\rho-1)^2 \leq 4\epsilon \end{aligned} \tag{12}$$

and bound $\Delta_2$ as follows,

$$\begin{aligned}\Delta_2 =& \left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right| = \left|\frac{(\rho-\alpha p_2 - (1-\alpha)\rho p_2)^2}{1-p_2} - \frac{(\rho-(\alpha-\epsilon)p_1 - (1-(\alpha-\epsilon))\rho p_1)^2}{1-p_1}\right| \\ \leq& \rho^2\left|p_1-p_2\right| + 2\alpha\left|\rho(\rho-1)\right|\left|p_2-p_1\right| + \alpha^2(1-\rho)^2\left|\frac{p_2^2}{1-p_2} - \frac{p_1^2}{1-p_1}\right| + \\ & \frac{2\epsilon p_1(1-\rho)\left|\rho-\alpha p_1-(1-\alpha)\rho p_1\right|}{1-p_1} + \frac{\epsilon^2 p_1^2(1-\rho)^2}{1-p_1} \leq 8.5\epsilon \end{aligned} \tag{13}$$

Combining the bounds in (7), (12) and (13) the error for in each cell in $R_{a,l}$ is bounded by $4(4\epsilon + 8.5\epsilon) = 50\epsilon$.

### C.6. Fourth region $R_{a,r}$

Let $R_{a,r} = \{(p,q)|\rho \leq p < 1, \rho p \leq q \leq \rho\}$. Clearly $R_{a,r} \subseteq R_a$, and by Lemma C.1 we know that WGI is increasing in $p$ and decreasing in $q$ in $R_{a,r}$.

We are now ready to define the cells. The cells will have two parameters $p_c \in [1-\rho, 1-\epsilon)$ and $\alpha \in [\epsilon, 1]$. Let

$$cell_{R_{a,r}}(p_c, \alpha) = \{(p,q)|p_c \leq p \leq p_c + \epsilon, (\alpha-\epsilon)\rho + (1-\alpha+\epsilon)\rho p \leq q \leq \alpha\rho + (1-\alpha)\rho p\} \ .$$

The cells would have $p_c = i\epsilon$, for $\rho/\epsilon \leq i \leq 1/\epsilon - 1$ and $\alpha = j\epsilon$, for $1 \leq j \leq 1/\epsilon$. This implies that there are $O(\epsilon^{-2})$ cells in $R_{a,r}$. Since in $R_{a,r}$ we have $\frac{\partial WGI}{\partial q} \leq 0$, there are $p_1, p_2 \in [1-\rho, 1)$, such that the maximum WGI value in $cell_{R_{a,r}}(p_c, \alpha)$ is obtained at a point $(p_1, q_1) = (p_1, (\alpha-\epsilon)\rho + (1-\alpha+\epsilon)\rho p_1)$ and the minimum value at $(p_2, q_2) = (p_2, \alpha\rho + (1-\alpha)\rho p_2)$

We bound $\Delta_1$ as follows,

$$\begin{aligned}
\Delta_1 =& \left|\frac{q_2^2}{p_2} - \frac{q_1^2}{p_1}\right| = \left|\frac{(\alpha\rho + (1-\alpha)\rho p_2)^2}{p_2} - \frac{((\alpha-\epsilon)\rho + (1-\alpha+\epsilon)\rho p_1)^2}{p_1}\right| \\
\leq& \rho^2\left(|p_2 - p_1| + 2\alpha\,|p_1 - p_2| + 2\epsilon\,|1-p_1|\right) + \rho^2\left|\frac{\alpha^2(1-p_2)^2}{p_2} - \frac{\alpha^2(1-p_1)^2}{p_1} - \frac{\epsilon(\epsilon-2\alpha)(1-p_1)^2}{p_1}\right| \\
\leq& 8\epsilon \qquad (14)
\end{aligned}$$

and $\Delta_2$ as follows,

$$\begin{aligned}
\Delta_2 =& \left|\frac{(\rho-q_2)^2}{1-p_2} - \frac{(\rho-q_1)^2}{1-p_1}\right| = \left|\frac{(\rho - \alpha\rho - (1-\alpha)\rho p_2)^2}{1-p_2} - \frac{(\rho - (\alpha-\epsilon)\rho - (1-(\alpha-\epsilon))\rho p_1)^2}{1-p_1}\right| \\
\leq& \rho^2(1-\alpha)^2\,|p_1 - p_2| + \rho^2\epsilon\,|2 - 2\alpha + \epsilon|\,(1-p_1) \leq 3\epsilon \qquad (15)
\end{aligned}$$

Combining the bounds in (7), (14) and (15) the error for in each cell in $R_{a,r}$ is bounded by $4(8\epsilon + 3\epsilon) = 44\epsilon$.

### C.7. WGI is continuous in $p = 0$ and $p = 1$

We show that the value of WGI is continuous at both extreme points $p = 0$ and $p = 1$ for point $(p,q) \in R$. Namely, we show that its value is $4\rho(1-\rho)$. This will allow us to extend the sub-domains from $p \in (0,1)$ to $p \in [0,1]$. Recall that,

$$GWI(p,q) = 4\left(\rho - \frac{q^2}{p} - \frac{(\rho-q)^2}{1-p}\right)$$

Consider $p \in (0, \delta]$ for $\delta < 1/2$. Since $(p,q) \in R$ this implies also that $q \leq p \leq \delta$. Therefore,

$$0 \leq \frac{q^2}{p} \leq \delta$$

We also have that

$$\rho^2 - 6\delta \leq \frac{(\rho-\delta)^2}{1-\delta} \leq \rho^2$$

the upper bound follows since the function is decreasing in $\delta$. For the lower bound, we have that $1/(1-\delta) < 1+2\delta$ for $\delta < 1/2$. This implies the following:

**Lemma C.2** *Let $(p,q) \in R$ such that $p \leq \delta \leq 1/2$. Then*

$$|GWI(p,q) - 4\rho(1-\rho)| \leq 6\delta$$

For $p \approx 1$ we have that for point $(p,q) \in R$ that $p-(1-\rho) \leq q \leq \min(p,\rho)$ and hence $q \approx \rho$. Similarly we show,

**Lemma C.3** *Let $(p,q) \in R$ such that $1-\delta \leq p \leq 1$, where $\delta \leq 1/2$. Then*

$$|GWI(p,q) - 4\rho(1-\rho)| \leq 3\delta$$

## Appendix D. Solution for the dependent case

In this section we are going to provide a minimal solution for the WGI in the dependent case.

**Lemma D.1** *Let $\boldsymbol{a} = D_v^\top X$ and $\boldsymbol{b} = (D_v \odot y)^\top X$, and assume $\boldsymbol{b} = \lambda \boldsymbol{a}$. The solutions for $\min(WGI_w)$ are $w_{1,2} = \pm\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$*

**Proof** Recall that according to 6:

$$WGI_w = WGI(P_w, Q_w) = 4\left(\rho - \frac{Q_w^2}{P_w} - \frac{(\rho - Q_w)^2}{1-P_w}\right)$$

and that $P_w = \frac{\boldsymbol{a}^\top w+1}{2}$ and $Q_w = \frac{\boldsymbol{b}^\top w+\rho}{2}$. Denote $x = \boldsymbol{a}^\top w$ and therefore $\boldsymbol{b}^\top w = \lambda \boldsymbol{a}^\top w = \lambda x$ we find the solution for:

$$\begin{aligned} g(x) =& 4\rho - \frac{4\left(\frac{\lambda x+\rho}{2}\right)^2}{\frac{x+1}{2}} - \frac{4\left(\rho - \frac{\lambda x+\rho}{2}\right)^2}{1-\frac{x+1}{2}} = 4\rho - \frac{2(\rho+\lambda x)^2}{1+x} - \frac{2(\rho-\lambda x)^2}{1-x} \\ =& 4\rho - \frac{2\left(\rho^2 + \lambda(\lambda-2\rho)x^2\right)}{1-x^2} \end{aligned}$$

We substitute $z = x^2$ and derive:

$$\frac{\partial g}{\partial z} = -2\frac{\lambda(\lambda-2\rho)(1-z) + (\rho^2 + \lambda(\lambda-2\rho)z)}{(1-z)^2} = -2 \cdot \frac{(\lambda-\rho)^2}{(1-z)^2}$$

by the chain rule:

$$\frac{\partial g}{\partial x} = \frac{\partial g}{\partial z} \cdot \frac{\partial z}{\partial x} = \frac{-4x(\lambda-\rho)^2}{(1-x^2)^2}$$

First we note that $x=0$ is an extreme point. Next, since $\frac{4(\lambda-\rho)^2}{(1-x^2)^2} > 0$, if $x > 0$ then $g$ is decreasing, and if $x < 0$ then $g$ is increasing. Therefore, the minimal points are at the edges of the feasible interval of $x$.

Since $x = \boldsymbol{a}^\top w$ and $\|w\| \leq 1$, according to $\cdot$ operator we have: $-\|\boldsymbol{a}\| \leq x \leq \|\boldsymbol{a}\|$. If $w = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$ then $x = \|a\|$, and if $w = -\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$ then $x = -\|a\|$. Therefore, $w_{1,2} = \pm\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$ are the solutions for this case. Clearly, both solutions attain the same value of WGI. ■

## Appendix E. Maximizing Information Gain for $P_w$ and $Q_w$

In this appendix we characterize various optimization problems that search for a feasible stochastic linear classifiers. The optimization problems depend on the information we assume about $P_w$ and $Q_w$, and their solutions are used in the proposed algorithms. We will assume that the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent throughout the appendix.

We first discuss the basic problem where the value of both $P_w$ and $Q_w$ are given (Appendix E.1). In Appendix E.2 we then show, that the optimal solution within a cell (as define in Appendix C) is found in one of the corners of that cell. The solution is described in Appendix E.1 and E.2 and used in algorithm `FixedPQ`. Next, in Appendix E.3 we solve the optimization used in `FindPSearchQ` for finding a feasible stochastic linear classifier where only the value of $P_w$ is given.

Finally, in Appendix E.4, we show that the set of feasible stochastic linear classifiers restricts the domain of WGI where solutions can be found.

### E.1. Values of $P_w$ and $Q_w$ are given

In this section we assume that the values of both $P_w$ and $Q_w$ are given, and we would like to compute whether there exists a feasible $w$, namely, $\|w\| = 1$, that attains those values. Such a $w$ implies a feasible stochastic linear classifier. Let the values be: $P_w = p$ and $Q_w = q$. The following lemma characterizes the solution.

**Lemma E.1** *Let $\boldsymbol{a} = D_v^\top X$ and $\boldsymbol{b} = (D_v \odot y)^\top X$, and assume that they are linearly independent. For the quadratic optimization problem:*

$$\begin{aligned} w^* = \arg\min \ & \|w\|_2^2 \\ & P_w = p \\ & Q_w = q \end{aligned}$$

*We have that,*

$$w^* = \frac{(2q-\rho)(\boldsymbol{a}\cdot\boldsymbol{b}) - (2p-1)\|\boldsymbol{b}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a}\cdot\boldsymbol{b})^2}\boldsymbol{a} + \frac{(2p-1)(\boldsymbol{a}\cdot\boldsymbol{b}) - (2q-\rho)\|\boldsymbol{a}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a}\cdot\boldsymbol{b})^2}\boldsymbol{b}$$

We will use Lemma E.1 in the following way. If for a given $(p, q)$ we have that resulting $w^*$ has $\|w^*\| \leq 1$, then we found a feasible stochastic linear classifier. Otherwise, we can conclude that there is no feasible stochastic linear classifier for $(p, q)$, since $w^*$ minimizes the norm.

**Proof** Recall that

$$P_w = 0.5 \cdot D_v^\top X w + 0.5 = p \quad \text{and} \quad Q_w = 0.5 \cdot (D_v \odot y)^\top X w + 0.5\rho = q$$

We define $\boldsymbol{a}^\top w = 2p - 1 = \bar{p}$ and $\boldsymbol{b}^\top w = 2q - \rho = \bar{q}$. With this notation we have

$$\begin{aligned} \min\ & 0.5 \sum_{i=1}^{d} w_i^2 \\ & \boldsymbol{a}^\top \cdot w - \bar{p} = 0 \\ & \boldsymbol{b}^\top \cdot w - \bar{q} = 0 \end{aligned}$$

We solve the optimization using the Lagrange multipliers:

$$L(w, \lambda_1, \lambda_2) = 0.5 \sum_{i=1}^{d} w_i^2 + \lambda_1(\boldsymbol{a}^\top w - \bar{p}) + \lambda_2(\boldsymbol{b}^\top w - \bar{q})$$

Considering the derivatives of the Lagrangian, we have

$$\frac{\partial L}{\partial w_i} = w_i + \lambda_1 \boldsymbol{a}_i + \lambda_2 \boldsymbol{b}_i = 0 \tag{16}$$

which implies that $\boldsymbol{w} = -\lambda_1 \boldsymbol{a} - \lambda_2 \boldsymbol{b}$. Also,

$$\frac{\partial L}{\partial \lambda_1} = \boldsymbol{a}^\top w - \bar{p} = 0 \tag{17}$$

Using that $\boldsymbol{w} = -\lambda_1 \boldsymbol{a} - \lambda_2 \boldsymbol{b}$ we have that $-\lambda_1 \|\boldsymbol{a}\|^2 - \lambda_2(\boldsymbol{a} \cdot \boldsymbol{b}) - \bar{p} = 0$ which implies that $\lambda_1 = -\frac{\lambda_2(\boldsymbol{a}\cdot\boldsymbol{b})+\bar{p}}{\|\boldsymbol{a}\|^2}$.[1] We consider the other derivative,

$$\frac{\partial L}{\partial \lambda_2} = \boldsymbol{b}^\top w - \bar{q} = 0 \tag{18}$$

Similarly, this implies that $-\lambda_1(\boldsymbol{a} \cdot \boldsymbol{b}) - \lambda_2 \|\boldsymbol{b}\|^2 - \bar{q} = 0$. Solving for $\lambda_1$ and $\lambda_2$ and $w^*$ we get,

$$\begin{aligned}
\lambda_2 =& \frac{\bar{p}(\boldsymbol{a} \cdot \boldsymbol{b}) - \bar{q}\|\boldsymbol{a}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2} \\
\lambda_1 =& \frac{\bar{q}(\boldsymbol{a} \cdot \boldsymbol{b}) - \bar{p}\|\boldsymbol{b}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2} \\
w^* =& \frac{\bar{q}(\boldsymbol{a} \cdot \boldsymbol{b}) - \bar{p}\|\boldsymbol{b}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2}\boldsymbol{a} + \frac{\bar{p}(\boldsymbol{a} \cdot \boldsymbol{b}) - \bar{q}\|\boldsymbol{a}\|^2}{\|\boldsymbol{a}\|^2\|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2}\boldsymbol{b}
\end{aligned} \tag{19}$$

Notice that this solution is valid only when the denominator is not 0. Therefore, it is valid if the $\boldsymbol{a}$ and $\boldsymbol{b}$ vectors are not in the same direction, i.e., $\forall \lambda : \boldsymbol{a} \neq \lambda \boldsymbol{b}$. ■

### E.2. Values of $P_w$ and $Q_w$ are within specific ranges

In this section we consider finding a feasible stochastic linear classifier inside a certain cell of values of $P_w$ and $Q_w$ (see Appendix C). The cell has the parameters $p_c$ and $\alpha$, in addition to $\epsilon \in (0, 1)$. Let $U(P_w, \rho)$ and $L(P_w, \rho)$ be linear functions which bound the values of $Q_w$ in the cell from above and below, respectively. The exact linear function depend on the sub-domain we are considering, namely the upper and lower boundaries of the region. Formally we solve the following quadratic optimization problem:

$$\begin{aligned}
&\min \|w\|_2^2 \\
&\quad p \leq P_w \leq p + \epsilon \\
&\quad (\alpha - \epsilon)U(P_w, \rho) + (1 - \alpha + \epsilon)L(P_w, \rho) \leq Q_w \leq \alpha U(P_w, \rho) + (1 - \alpha)L(P_w, \rho)
\end{aligned}$$

1. We note that $\|\boldsymbol{a}\| > 0$ since we assume a coordinate with a fixed positive value for every sample, to allow for a threshold.

Since we minimize the norm of $w$, if there exists a feasible stochastic linear classifier in the cell, the result of this optimization would be feasible. If there is no feasible stochastic linear classifier in this cell, the result of this optimization would be a non feasible $w$ (either $\|w\| > 1$ or no solution at all).

**Lemma E.2** *Let $\boldsymbol{a} = D_v^\top X$ and $\boldsymbol{b} = (D_v \odot y)^\top X$, and assume they are linearly independent. Let $\epsilon \in (0,1)$, $\alpha \in [\epsilon, 1]$ and $p \in [0, 1-\epsilon]$. Let $U(P_w, \rho)$ and $L(P_w, \rho)$ be linear functions, such that:*

$$\forall P_w \in [p, p+\epsilon] : \begin{cases} L(P_w,\rho) < U(P_w,\rho) & p \in (0,1) \\ L(P_w,\rho) = U(P_w,\rho) & p = 0 \text{ or } p = 1 \end{cases}$$

*Then the solution for:*

$$\begin{aligned} &\min \|w\|_2^2 \\ &\quad p \le P_w \le p+\epsilon \\ &\quad (\alpha-\epsilon)U(P_w,\rho) + (1-\alpha+\epsilon)L(P_w,\rho) \le Q_w \le \alpha U(P_w,\rho) + (1-\alpha)L(P_w,\rho) \end{aligned}$$

*coincides with a solution of*

$$\begin{aligned} &\min \|w\|_2^2 \\ &\quad P_w = p' \\ &\quad Q_w = q' \end{aligned}$$

*for $p' \in \{p, p+\epsilon\}$ and $q' \in \{(\alpha-\epsilon)U(P_w,\rho)+(1-\alpha+\epsilon)L(P_w,\rho), \alpha U(P_w,\rho)+(1-\alpha)L(P_w,\rho)\}$*

We note that in order to compute the actual value, one could use the solution of Lemma E.1 in order to evaluate each of the four combinations for $(p', q')$. The above lemma guarantees that the optimal solution indeed coincides in one of the corners.

**Proof** The above optimization is equivalent to the following optimization problem:

$$\begin{aligned} &\min 0.5 \sum_{i=1}^{d} w_i^2 \\ &\overbrace{p - P_w}^{h_I} \le 0 \\ &\overbrace{P_w - (p+\epsilon)}^{h_{II}} \le 0 \\ &\overbrace{(\alpha-\epsilon)U(P_w,\rho) + (1-\alpha+\epsilon)L(P_w,\rho) - Q_w}^{h_{III}} \le 0 \\ &\overbrace{Q_w - \alpha U(P_w,\rho) - (1-\alpha)L(P_w,\rho)}^{h_{IV}} \le 0 \end{aligned}$$

Since $P_w$, $Q_w$, $U$ and $L$ are all linear functions of $w$ with the constants $X$, $y$, $D_l$ and $\rho$, this formulation corresponds to an optimization problem with a quadratic target and linear constraints. We solve using the Lagrange multipliers:

$$L(w, \lambda_1, \lambda_2, \mu_1, \mu_2) = 0.5 \cdot \sum_{i=1}^{d} w_i^2 + \lambda_1 h_I + \lambda_2 h_{II} + \mu_1 h_{III} + \mu_2 h_{IV}$$

We want to minimize $L$ s.t $\lambda_1, \lambda_2, \mu_1, \mu_2 \geq 0$.

First we consider the pair $h_I$ and $h_{II}$; if both constraints are summed the result is,

$$p - P_w + P_w - (p + \epsilon) = -\epsilon < 0$$

Therefore at least one of the constraints is strictly negative (otherwise their sum cannot be negative). Since $L$ is to be minimized, for every $w$ the solution $(w, \lambda_1, \lambda_2...)$, is going to select at least one of $\lambda_1 > 0$ or $\lambda_2 > 0$ in order to use the negative term and reach a smaller value of $L$. According to the complementary slackness this means that either $P_w = p$ or $P_w = p + \epsilon$ (of course we cannot have both).

Next, we consider constraints $h_{III}$ and $h_{IV}$; we denote with $\Delta$ the sum between these two constraints:

$$\begin{aligned}\Delta =&(\alpha - \epsilon)U(P_w, \rho) + (1 - \alpha + \epsilon)L(P_w, \rho) - Q_w + Q_w - \alpha U(P_w, \rho) - (1 - \alpha)L(P_w, \rho)\\ =&\epsilon\left(L(P_w, \rho) - U(P_w, \rho)\right)\end{aligned}$$

Since $\epsilon > 0$ we only need to consider the term $(L(P_w, \rho) - U(P_w, \rho))$. We have three cases: $P_w = 0$, $P_w = 1$ and $0 < P_w < 1$.

For the cases $P_w = 0$ and $P_w = 1$, according to the definition of $U$ and $L$ $(L(0, \rho) - U(0, \rho)) = 0$ or $(L(1, \rho) - U(1, \rho)) = 0$ correspondingly. For both $\Delta = 0$, and both $h_{III}$ and $h_{IV}$ collapse to $Q_w = 0$ or $\rho$. Therefore the solution is attained either at $(0, 0)$ for when $P_w = 0$ or at $(1, \rho)$ for when $P_w = 1$.

Next, since for $0 < P_w < 1$, $L(P_w, \rho) - U(P_w, \rho) < 0$ and $\epsilon > 0$ therefore $\Delta < 0$. Similarly to the $P_w$ constraints, according to the complementary slackness, this means that either $h_{III}$ or $h_{IV}$ is tight.

Therefore each minimal norm solution occurs when exactly one of $h_I$ or $h_{II}$ is tight, and when one of $h_{III}$ or $h_{IV}$ is tight. In each of those four cases, the solution occurs in one of the corners of the cell, which in turn is the same as solving the optimization problem for constant values of $P_w$ and $Q_w$ for each of the corners of the cell and taking the best solution. ■

### E.3. Given value $P_w$ search over $Q_w$

This section takes a different approach then the previous ones. We search for a feasible stochastic linear classifier given a value for only $P_w$. Instead of minimizing the norm of the classifier, we search for $Q_w$ values that minimize the WGI under the constraint that the solution is feasible.

The following lemma characterizes the solution when we minimize directly WGI for a given $P_w$. (Note that when we fix the value of $P_w$ the function WGI is convex.)

**Lemma E.3** *Let $\boldsymbol{a} = D_v^\top X$ and $\boldsymbol{b} = (D_v \odot y)^\top X$, and assume that they are linearly independent. Let $\bar{\boldsymbol{b}} = \boldsymbol{b} - \frac{(\boldsymbol{a}\cdot\boldsymbol{b})}{\|\boldsymbol{a}\|^2}\boldsymbol{a}$. Then for:*

$$\begin{aligned}w^* = \arg\min\ & WGI(p, Q_w)\\ & P_w = p\end{aligned}$$

*then the two solutions are,*

$$w^* = \frac{2p-1}{\|\boldsymbol{a}\|^2}\boldsymbol{a} \pm \frac{\sqrt{\|\boldsymbol{a}\|^2 - (2p-1)^2}}{\|\boldsymbol{a}\|\|\bar{\boldsymbol{b}}\|}\bar{\boldsymbol{b}}$$

**Proof** The first observation we have is that for a fixed $P_w = p$, $q = \rho p$ is a maximum for WGI, and since the second derivative of WGI is negative, we get that WGI attains a minimal value when $Q_w$ is either maximized or minimized. Therefore we substitute the original problem with the following two optimization problems

$$Q^{\max} = \arg\max Q_w \quad \text{and} \quad Q^{\min} = \arg\min Q_w$$
$$P_w = p \qquad\qquad P_w = p$$
$$\|w\|^2 \leq 1 \qquad\qquad \|w\|^2 \leq 1$$

where the first constraint is used to fix $P_w$ and the second constraint is used to bound the norm of $w$. Now, we will translate the maximization while using the previous expressions for $\boldsymbol{a}$, $\boldsymbol{b}$ and set $\bar{p} = 2p - 1$.

$$\begin{aligned} &\max \boldsymbol{b}^\top w \\ &\quad \boldsymbol{a}^\top w - \bar{p} = 0 \\ &\quad \|w\|^2 \leq 1 \end{aligned}$$

We rewrite $\boldsymbol{b} = \alpha\boldsymbol{a} + \bar{\boldsymbol{b}}$ as a sum of: (1) $\alpha \cdot \boldsymbol{a}$ which is $\boldsymbol{b}$'s projection over $\boldsymbol{a}$ and (2) $\bar{\boldsymbol{b}}$ which is a vector that is orthogonal to $a$, i.e., $(a \cdot \bar{b}) = 0$. (Since $a$ and $b$ are linearly independent we have $\bar{\boldsymbol{b}} \neq 0$.) We write $w$ as:

$$w = \mu_1\boldsymbol{a} + \mu_2\bar{\boldsymbol{b}} + \bar{\boldsymbol{w}}$$

where $\bar{\boldsymbol{w}}$ is orthogonal to $\boldsymbol{a}$ and $\bar{\boldsymbol{b}}$. The optimization becomes:

$$\begin{aligned} &\max \alpha\mu_1\|\boldsymbol{a}\|^2 + \mu_2\|\bar{\boldsymbol{b}}\|^2 \\ &\quad \mu_1\|\boldsymbol{a}\|^2 - \bar{p} = 0 \\ &\quad \mu_1^2\|\boldsymbol{a}\|^2 + \mu_2^2\|\bar{\boldsymbol{b}}\|^2 + \|\bar{\boldsymbol{w}}\|^2 \leq 1 \end{aligned}$$

When we substitute $\mu_1 = \frac{\bar{p}}{\|\boldsymbol{a}\|^2}$ we get:

$$\begin{aligned} &\max \alpha\bar{p} + \mu_2\|\bar{\boldsymbol{b}}\|^2 \\ &\quad \frac{\bar{p}^2}{\|\boldsymbol{a}\|^2} + \mu_2^2\|\bar{\boldsymbol{b}}\|^2 + \|\bar{\boldsymbol{w}}\|^2 \leq 1 \end{aligned}$$

Since $\alpha\bar{p}$ is a constant and $\|\bar{\boldsymbol{b}}\|^2$ is positive, the optimization is equivalent to:

$$\begin{aligned} &\max \mu_2 \\ &\quad \mu_2^2\|\bar{\boldsymbol{b}}\|^2 + \|\bar{\boldsymbol{w}}\|^2 \leq 1 - \frac{\bar{p}^2}{\|\boldsymbol{a}\|^2} \end{aligned}$$

Notice that since only $\mu_2$ is a variable that contributes to the max, we would like to make it as large as possible under the constraint. Therefore the optimal solution would be when the $\bar{w} = 0$, and the expression of $\mu_2$ would be:

$$\mu_2^2\|\bar{\boldsymbol{b}}\|^2 = 1 - \frac{\bar{p}^2}{\|\boldsymbol{a}\|^2} \quad \Rightarrow \quad \mu_2 = \pm\frac{\sqrt{\|\boldsymbol{a}\|^2 - \bar{p}^2}}{\|\boldsymbol{a}\|\|\bar{\boldsymbol{b}}\|}$$

The maximal solution for the optimization is achieved with a positive $\mu_2$ in the above expression:

$$\mu_2 = \frac{\sqrt{\|\boldsymbol{a}\|^2 - \bar{p}^2}}{\|\boldsymbol{a}\|\|\bar{\boldsymbol{b}}\|}$$

Finally the solution for $w$ is,

$$w = \frac{\bar{p}}{\|\boldsymbol{a}\|^2}a + \frac{\sqrt{\|\boldsymbol{a}\|^2 - \bar{p}^2}}{\|\boldsymbol{a}\|\|\bar{\boldsymbol{b}}\|}\bar{\boldsymbol{b}}$$

Notice that for the minimization problem $\min Q_w$ we simply select the negative value of $\mu_2$, and get,

$$w = \frac{\bar{p}}{\|\boldsymbol{a}\|^2}\boldsymbol{a} - \frac{\sqrt{\|\boldsymbol{a}\|^2 - \bar{p}^2}}{\|\boldsymbol{a}\|\|\bar{\boldsymbol{b}}\|}\bar{\boldsymbol{b}}$$

This implies that the solution is either one of:

$$w^*_{1,2} = \frac{2p-1}{\|\boldsymbol{a}\|^2}\boldsymbol{a} \pm \frac{\sqrt{\|\boldsymbol{a}\|^2 - (2p-1)^2}}{\|\boldsymbol{a}\|\|\bar{\boldsymbol{b}}\|}\bar{\boldsymbol{b}}$$

■

### E.4. Feasible Ranges for $P_w$ and $Q_w$

In all of our problems we have constraints such as $P_w = p$ and $Q_w = q$ (or their range variants). However we should note that not all possible values of $p$ and $q$ are applicable for a given data:

**Lemma E.4** *Let $\boldsymbol{a} = D_v^\top X$ and $\boldsymbol{b} = (D_v \odot y)^\top X$ and let $w$ be a feasible stochastic linear classifier. Then:*

$$0.5 - 0.5\|\boldsymbol{a}\| \leq P_w \leq 0.5 + 0.5\|\boldsymbol{a}\|$$
$$0.5\rho - 0.5\|\boldsymbol{b}\| \leq Q_w \leq 0.5\rho + 0.5\|\boldsymbol{b}\|$$

**Proof** First:

$$-\|\boldsymbol{a}\|\|w\| \leq \boldsymbol{a}^\top \cdot w \leq \|\boldsymbol{a}\|\|w\| \rightarrow -\|\boldsymbol{a}\| \leq \boldsymbol{a}^\top \cdot w \leq \|\boldsymbol{a}\|$$
$$-\|\boldsymbol{b}\|\|w\| \leq \boldsymbol{b}^\top \cdot w \leq \|\boldsymbol{b}\|\|w\| \rightarrow -\|\boldsymbol{b}\| \leq \boldsymbol{b}^\top \cdot w \leq \|\boldsymbol{b}\|$$

where the left-hand side is a property of $\cdot$, and the right-hand is because $w$ is feasible. Applying the above inequalities in expression 5: $P_w = 0.5\boldsymbol{a}^\top \cdot w + 0.5$ and $Q_w = 0.5\boldsymbol{b}^\top \cdot w + 0.5\rho$ concludes the proof.

■

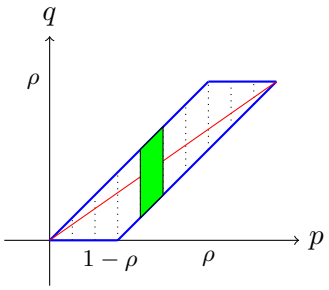


Figure 6: Partition $P_w$ to slices for $\rho \geq 0.5$

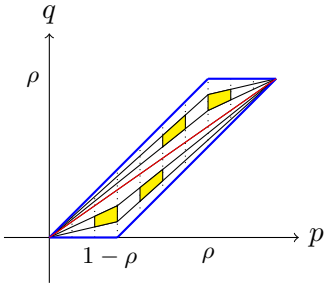


Figure 7: Partition $(P_w, Q_w)$ in to trapezoid cells for $\rho \geq 0.5$

## Appendix F. Approximation Algorithms

This section details the three algorithms discussed in theorem 6.2:

**Theorem 6.2** *Let $S$ be a sample, $D_v$ be a distribution over $S$, and $\epsilon$ be an approximation parameter. Let $\boldsymbol{a} = D_v^\top X$ and $\boldsymbol{b} = (D_v \odot y)^\top X$. Then: For the case where $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent, algorithms* `FixedPQ` *and* `FixedPSearchQ` *guarantee an $\epsilon$-approximation for $WGI$ using a stochastic linear classifier. Algorithm* `FixedPQ` *runs in time $O(Nd) + O\left(\epsilon^{-2}\log\left(\epsilon^{-1}\right)\right)$, and algorithm* `FixedPSearchQ` *runs in time $O(Nd) + O\left(\frac{d}{\epsilon}\right)$. And for the case where $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, i.e., $\boldsymbol{b} = \lambda\boldsymbol{a}$, algorithm* `DependentWGI` *runs in time $O(Nd)$, and achieves the optimal result for WGI using a stochastic linear classifier.*

For each algorithm we provide pseudo code, correctness proof and a complexity analysis: Algorithm `FixedPQ` is in Appendix F.1, Algorithm `FixedPSearchQ` is in Appendix F.2 and Algorithm `DependentWGI` is in Appendix F.3.

As mentioned all three algorithms depend on $\boldsymbol{a}$ the weighted average of the data, and $\boldsymbol{b}$ the weighted average of the positive class. We note here that both $\boldsymbol{a}$ and $\boldsymbol{b}$, along with other static related computations such as $\|\boldsymbol{a}\|, \|\boldsymbol{b}\|$ are treated as global variables which are computed once in the beginning of each algorithm. We also note that the constant $C = 100$ used in the pseudo code corresponds to the bound from Appendix C.

### F.1. `FixedPQ` Approximate minimal Gini Index in cells of $P_w, Q_w$

In this section we describe Algorithm `FixedPQ`. After computing the expressions as described earlier, we generate a list *candidates* that contains pairs $(p, q)$ which are corners of the trapezoid cells:

Initially we generate a set of equally spaces $P_w$ values from the interval $[\max(0, 0.5(1 - \|\boldsymbol{a}\|)), \min(1, 0.5(1 + \|\boldsymbol{a}\|))]$. This interval is the intersection of the domain of WGI and the feasible range of $w$ as described in Appendix E.4. The set of $P_w$ values also includes the values for $\rho$ and $1 - \rho$ in order to ensure that the candidates partition the domain according to the regions defined in Appendix C. Figure 7 shows the partition to trapezoid cells.

Next, for each $P_w$ value we define equally spaced values in ranges $[\max(0, p + \rho - 1, 0.5(\rho - \|\boldsymbol{b}\|)), \min(\rho p, 0.5(\rho + \|\boldsymbol{b}\|))]$ and $[\max(\rho p, 0.5(\rho - \|\boldsymbol{b}\|)), \min(\rho, p, 0.5(\rho + \|\boldsymbol{b}\|))]$. Again, these ranges

take into consideration both the sub-domains of the WGI and the feasible ranges of the stochastic linear classifier.

The equally spaced values are generated using the $GetRange$ function, and we always partition into $\frac{C}{\epsilon}$ values in order to achieve an approximation of $\epsilon$ (see correctness below). This results in $O(\epsilon^{-1})$ values of $P_w$, and for each of those $O(\epsilon^{-1})$ for the $Q_w$ values below and above $Q_w = \rho P_w$. The total number of candidates is $O(\epsilon^{-2})$.

The list $candidates$ now contains pairs of $(P_w, Q_w)$ which are sorted according to WGI in an ascending order. For each point made of $P_w$ and $Q_w$ values will try to fit a stochastic linear classifier: First we use the function $GetCandidateWeightsTerms$ to get the scalars $\alpha$ and $\beta$. Since we also know $\|\boldsymbol{a}\|$, $\|\boldsymbol{b}\|$ and $(\boldsymbol{a} \cdot \boldsymbol{b})$ we measure the feasibility of the classifier $w$ before computing it. Once we find a feasible classifier, we compute it explicitly and return it.

**Algorithm correctness** Denote $w^*$ the optimal feasible solution with respect to WGI: $WGI_{w^*} = \min_{\{w|\|w\|\leq 1\}} WGI_w$. Let $Cells$ be the union of non-intersecting trapezoid cells parametrized by step size $\epsilon/C$. We denote $c^* \in Cells$ the cell which contains $w^*$. Denote $W_c$ the feasible solutions proposed to cell $c \in Cells$ by the optimization in Appendix E.2 (since $w^*$ is feasible, we know that at least one corner of cell $c$ is feasible). Specifically for $c^*$, we denote the best proposed solution as $w_1$:

$$w_1 = \arg\min_{w \in W_{c^*}} (WGI(w))$$

Since both $w^*, w_1 \in c^*$ according to the bound we get $WGI_{w^*} \leq WGI_{w_1} \leq WGI_{w^*} + \epsilon$.

Next, we define $W_{Cells}$ as the union of the solutions from all the cells: $W_{Cells} = \bigcup_{c \in Cells} W_c$. $W_{Cells}$ is the collection of all the classifiers that would have been created from evaluating every point in candidates. Since $W_{c^*} \subset W_{Cells}$ we have $w_1 \in W_{Cells}$. Let $w_2$ be the minimal solution from all the solutions in $W_{Cells}$:

$$w_2 = \arg\min_{w \in W_{Cells}} (WGI_w)$$

In particular, $WGI_{w^*} \leq WGI_{w_2} \leq WGI_{w_1} \leq WGI_{w^*} + \epsilon$, and we notice that $w_2$ is actually the result returned by the algorithm. We therefore conclude that the algorithm returns an $\epsilon$ approximation to the optimal WGI.

**Time complexity** We start by computing $\rho$ which consists of dot-product over vectors of dimension $N$ and takes time $O(N)$. Next, we calculate $\boldsymbol{a}$, $\boldsymbol{b}$, by multiplying an $N$ dimensional vector by an $N \times O(d)$ matrix, which is done in time $O(Nd)$. Computing the expressions that are used often takes time $O(d)$.

We create a list $candidates$, consisting of $O(\epsilon^{-2})$ pairs $(p, q)$ in time $O(\epsilon^{-2})$. Next, we compute the WGI for each candidate $(p, q)$ in $O(1)$ time, and we sort them using the WGI values in time $O\left(\epsilon^{-2} \log\left(\epsilon^{-2}\right)\right) = O\left(\epsilon^{-2} \cdot \log\left(\epsilon^{-1}\right)\right)$.

Finally we try to match a classifier for each point: we iterate over (sorted) $candidates$, and we call the function $GetCandidateWeightsTerms$ which runs in time $O(1)$ since it is using the pre-calculated expressions. There are at most $O\left(\epsilon^{-2}\right)$ items in the list. Therefore this step costs $O\left(\epsilon^{-2}\right)$. Once a solution is found to be feasible , only then do we compute the weight vector in time $O(d)$.

**Algorithm** `FixedPQ` ($X$ - *dataset,* $D_v$ - *data distribution,* $y$ - *labels,* $\epsilon$ - *approximation range*)

Store globally: $\rho \leftarrow D_v \cdot y, \boldsymbol{a}^\top \leftarrow D_v^t X, \boldsymbol{b}^\top \leftarrow (D_v \odot y)^\top X$
Compute once and store globally: $\|\boldsymbol{a}\|, \|\boldsymbol{b}\|, (\boldsymbol{a} \cdot \boldsymbol{b})$
candidates $\leftarrow$ new List()
**foreach** $p \in$`GetRange` *(*$\max(0, 0.5(1 - \|\boldsymbol{a}\|))$, $\min(1, 0.5(1 + \|\boldsymbol{a}\|))$, $\frac{C}{\epsilon}$*)*$\bigcup\{\rho, 1-\rho\}$ **do**
  **foreach** $q \in$`GetRange` *(*$\max(0, p + \rho - 1, 0.5(\rho - \|\boldsymbol{b}\|))$, $\min(\rho p, 0.5(\rho + \|\boldsymbol{b}\|))$, $\frac{C}{\epsilon}$*)* **do**
    candidates.Insert($(p, q)$)
  **end**
  **foreach** $q \in$`GetRange` *(*$\max(\rho p, 0.5(\rho - \|\boldsymbol{b}\|))$, $\min(\rho, p, 0.5(\rho + \|\boldsymbol{b}\|))$, $\frac{C}{\epsilon}$*)* **do**
    candidates.Insert($(p, q)$)
  **end**
**end**
candidates $\leftarrow$ candidates.Sort(ascending=True, by = $WGI(p, q)$)
**foreach** $p, q \in$ *candidates* **do**
  $(\alpha, \beta) \leftarrow$ `GetCandidateWeightsTerms`*(p, q)*
  **if** $\alpha^2 \cdot \|\boldsymbol{a}\|^2 + 2\alpha\beta \cdot (\boldsymbol{a} \cdot \boldsymbol{b}) + \beta^2 \cdot \|\boldsymbol{b}\|^2 \leq 1$ **then**
    **return** $(\alpha \cdot \boldsymbol{a} + \beta \cdot \boldsymbol{b}, WGI(p, q))$
  **end**
**end**
**return** *"Error - Impossible"*

**Procedure** `GetCandidateWeightsTerms`(*p, q*)

$\bar{p} \leftarrow 2p - 1$ , $\bar{q} \leftarrow 2q - \rho$
**return** $\left( \frac{\bar{q} \cdot (\boldsymbol{a} \cdot \boldsymbol{b}) - \bar{p} \cdot \|\boldsymbol{b}\|^2}{\|\boldsymbol{a}\|^2 \cdot \|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2}, \frac{\bar{p} \cdot (\boldsymbol{a} \cdot \boldsymbol{b}) - \bar{q} \cdot \|\boldsymbol{a}\|^2}{\|\boldsymbol{a}\|^2 \cdot \|\boldsymbol{b}\|^2 - (\boldsymbol{a} \cdot \boldsymbol{b})^2} \right)$

**Procedure** `GetRange`(*lower, upper, steps*)

**return** $\{p |$ *equally spaced points in* $[$*lower,upper*$]$ *with step=*$\frac{\text{upper - lower}}{\text{steps}}\}$

**Algorithm 2:** Approximating optimal WGI with 2 ranges

**The final complexity is:**

$$O(Nd) + O(d) + O\left(\epsilon^{-2}\right) + O\left(\epsilon^{-2} \cdot \log\left(\epsilon^{-1}\right)\right) + O\left(\epsilon^{-2}\right) + O(d) = O(Nd) + O\left(\epsilon^{-2} \cdot \log\left(\epsilon^{-1}\right)\right)$$

### F.2. `FixedPSearchQ` Approximate minimal Gini Index in range of $P_w$

In this section we describe Algorithm `FixedPSearchQ`. After computing the global expressions as described earlier, we generate a list $candidates$ that contains values $p$ which are equally spaced values on the feasible range of $P_w$:

This set is generated from the interval $[\max(0, 0.5(1 - \|\boldsymbol{a}\|)), \min(1, 0.5(1 + \|\boldsymbol{a}\|))]$. This interval is the intersection of the domain of WGI and the feasible range of $w$ as described in Appendix E.4. The set of $P_w$ values also includes the values for $\rho$ and $1 - \rho$ in order to ensure that the candidates partition the domain according to the regions defined in Appendix C. Figure 6 shows the partition to slices of trapezoid cells.

The equally spaced values are generated using the $GetRange$ function, and we partition into $\frac{C}{\epsilon}$ values in order to achieve an approximation of $\epsilon$ (see correctness below). This results in $O(\epsilon^{-1})$ values of $P_w$.

Next, we compute $\bar{\boldsymbol{b}}$ which is $\boldsymbol{b}$ minus $\boldsymbol{a}$'s projection over it. This allows for the computation of $\boldsymbol{u}$ and $\boldsymbol{v}$. These vectors are components of the final solution.

Finally, for each $P_w$ we fit two solutions which are functions of $P_w$, $\boldsymbol{u}$ and $\boldsymbol{v}$. We keep the best result $\hat{w}$ and its matching WGI $\hat{g}$ using the function $SetCandidateIfBetter$, and return $\hat{w}$ and $\hat{g}$ at the end of the loop.

**Algorithm correctness** Given an $\epsilon$ parameter for the algorithm, we know that the optimal feasible solution $w^*$ belongs to some range of $P_w$ in the grid, denote that range as $C^* = [\alpha, \beta]$.

Denote $p^* = P_{w^*}, q^* = Q_{w^*}$. First we note that $p^* \in C^*$ (by definition), second, denote by $C_q$ the $Q$ range that would have included $w^*$ if we were to partition into trapezoid cells. Since $w^*$ is feasible, at least one of the corners of $C^* \times C_q$ is feasible. There is a feasible solution that we denote as $w_c$ at $\alpha$ or $\beta$. Without loss of generality assume it is $\alpha$. According to the bound on the WGI in Appendix C, this candidate solution is at most $\epsilon$ away from $w^*$. Since our algorithm considers all the possible feasible solutions for $\alpha$ it also considers $w_c$ and therefore the result of the optimization problem as presented in Appendix E.3 denoted as $w_1$ is at least as good as $w_c$:

$$WGI_{w_1} - WGI_{w^*} \leq WGI_{w_c} - WGI_{w^*} \leq \epsilon$$

Finally, if the algorithm returns a different solution $w_2$ it is only because $WGI_{w_2} \leq WGI_{w_1}$, and we have

$$WGI_{w^*} \leq WGI_{w_2} \leq WGI_{w_1} \leq WGI_{w^*} + \epsilon$$

namely, $w_2$ is also an $\epsilon$ approximation.

**Time complexity** We start by calculating $\rho$ which consists of dot-product of vectors of dimension $N$ and done in time $O(N)$. Next, we calculate $\boldsymbol{a}$ an $\boldsymbol{b}$, by multiplying an $N$ dimensional vector by an $N \times O(d)$ matrix in time $O(Nd)$. We calculate expressions that are used often, each in time $O(d)$.

Finally the loop iterates over $O(\epsilon^{-1})$ values of $p$. For each such value, the algorithm computes a weight vector and uses the method $SetCandidateIfBetter$ which has time complexity $O(d)$. The total cost of all the iterations is $O(\frac{d}{\epsilon})$.

**Algorithm** `FixedPSearchQ(`$X$ *- dataset,* $D_v$ *- data distribution,* $y$ *- labels,* $\epsilon$ *- approximation range*`)`

Store globally: $\rho \leftarrow D_v \cdot y, \boldsymbol{a}^\top \leftarrow D_v^t X, \boldsymbol{b}^\top \leftarrow (D_v \odot y)^\top X$
Compute once and store globally: $\|\boldsymbol{a}\|, \|\boldsymbol{b}\|, (\boldsymbol{a} \cdot \boldsymbol{b})$
$\bar{\boldsymbol{b}} \leftarrow \boldsymbol{b} - \frac{(\boldsymbol{a} \cdot \boldsymbol{b})}{\|\boldsymbol{a}\|^2} \cdot \boldsymbol{a}$
$\boldsymbol{u} = \frac{1}{\|\boldsymbol{a}\|^2} \cdot \boldsymbol{a}$, $\boldsymbol{v} = \frac{1}{\|\boldsymbol{a}\|\|\bar{\boldsymbol{b}}\|} \cdot \bar{\boldsymbol{b}}$
$\hat{w} \leftarrow \perp, \hat{g} \leftarrow \perp$
**foreach** $p \in$`GetRange (`$\max(0, 0.5(1 - \|\boldsymbol{a}\|))$, $\min(1, 0.5(1 + \|\boldsymbol{a}\|))$, $\frac{C}{\epsilon}$`)`$\bigcup\{\rho, 1 - \rho\}$ **do**
  $\bar{p} \leftarrow 2p - 1$
  $w_1 \leftarrow \bar{p} \cdot \boldsymbol{u} + \sqrt{\|\boldsymbol{a}\|^2 - \bar{p}^2} \cdot \boldsymbol{v}$
  $\hat{w}, \hat{g} \leftarrow$ `SetCandidateIfBetter(`$w_1$,$\hat{w}$,$\hat{g}$,$p$`)`
  $w_2 \leftarrow \bar{p} \cdot \boldsymbol{u} - \sqrt{\|\boldsymbol{a}\|^2 - \bar{p}^2} \cdot \boldsymbol{v}$
  $\hat{w}, \hat{g} \leftarrow$ `SetCandidateIfBetter(`$w_2$,$\hat{w}$,$\hat{g}$,$p$`)`
**end**
**if** $\hat{w} = \perp$ **then**
  **return** *"Error - Impossible"*
**end**
**return** $\hat{w}, \hat{g}$

**Procedure** `SetCandidateIfBetter(`$w$,$\hat{w}$,$\hat{g}$,$p$`)`

$q \leftarrow \frac{\boldsymbol{b}^\top w + \rho}{2}$
**if** $WGI(p, q) \leq \hat{g}$ **then**
  **return** $w, WGI(p, q)$
**end**
**else**
  **return** $\hat{w}, \hat{g}$
**end**

**Procedure** `GetRange(`*lower, upper, steps*`)`

**return** $\{p|$ *equally spaced points in* $[$*lower,upper*$]$ *with step=*$\frac{\text{upper - lower}}{\text{steps}}\}$

**Algorithm 3:** Approximating optimal WGI with one range over $P_w$

**The final complexity is:**

$$O(Nd) + O(d) + O\left(\frac{d}{\epsilon}\right) = O(Nd) + O\left(\frac{d}{\epsilon}\right)$$

### F.3. `DependentWGI` Dependent constraints case

In this section we describe Algorithm `DependentWGI`. Since we have an analytical solution (Appendix D), we simply apply it and return its result as $w$ and the corresponding WGI $g$.

**Algorithm** `DependentWGI` ($X$ - *dataset,* $D_v$ - *data distribution,* $y$ - *labels*)

Store globally: $\rho \leftarrow D_v \cdot y, \boldsymbol{a}^\top \leftarrow D_v^t X, \boldsymbol{b}^\top \leftarrow (D_v \odot y)^\top X$
Compute once and store globally: $\|\boldsymbol{a}\|$
Get $\lambda$ s.t $\boldsymbol{b} = \lambda \boldsymbol{a}$ ▷ According to our assumption, this always happens
$w \leftarrow \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$
$g \leftarrow 4\left(\rho - \frac{(\rho+\lambda\|\boldsymbol{a}\|)^2}{2(1+\|\boldsymbol{a}\|)} - \frac{(\rho-\lambda\|\boldsymbol{a}\|)^2}{2(1-\|\boldsymbol{a}\|)}\right)$
**return** $w, g$

**Algorithm 4:** Approximating optimal WGI dependent case

**Algorithm correctness** See Appendix D.

**Algorithm complexity** We start by calculating $\rho$ which consists of dot-product over vectors of dimension $N$ in time $O(N)$. Next, we calculate $\boldsymbol{a}$ and $\boldsymbol{b}$, by multiplying an $N$ vector by an $N \times O(d)$ matrix in time $O(Nd)$. Also, we calculate $\|\boldsymbol{a}\|$ in time $O(d)$. And finally, we extract $\lambda$ which is $O(1)$ without verification (simply, $\lambda = \frac{\boldsymbol{b}_1}{\boldsymbol{a}_1}$), or $O(d)$ with verification (make sure that $\forall i \in [1, d] : \lambda = \frac{\boldsymbol{b}_i}{\boldsymbol{a}_i}$). We compute $w$ in time $O(d)$ and the drop in constant time.

The final complexity is:

$$O(Nd) + O(d) + O(d) = O(Nd)$$

## Appendix G. WGI not convex

In this section we are going to show that the WGI is non convex in $w$.

**Lemma G.1** *The function $WGI_w$ is non convex*

**Proof** Let $x_1, x_2 \in R^d$ be samples s.t $x_{1,1} = -x_{2,1} = 2x$ and $\forall i \in [2, d] : x_{1,i} = x_{2,i}$ ($x_1$ and $x_2$ are opposite in the first dimension, and the same in all other). Let $y_1 = 1$ be the label of $x_1$ and let $y_2 = 0$ be the label of $x_2$. Also let $D$ be the data distribution s.t $D(x_1) = D(x_2) = 0.5$. We note that this entails that $\rho = 0.5$ and that $\boldsymbol{a}_1 = 0$ while $\boldsymbol{b}_1 = x$.

We define three feasible stochastic linear classifiers (1) $w_0 = \mathbf{0}$ all zeros vector, (2) $w_1$ whose first coordinate is 1 and the rest 0: $w_{1,1} = 1, \forall i \in [2, d] : w_{1,i} = 0$ and (3) $w_m$ whose first coordinate is 0.5 and the rest 0: $w_{1,1} = 0.5, \forall i \in [2, d] : w_{1,i} = 0$. We see that

$$\boldsymbol{a}^\top w_0 = \boldsymbol{a}^\top w_1 = \boldsymbol{a}^\top w_m = 0$$
$$\boldsymbol{b}^\top w_0 = 0, \quad \boldsymbol{b}^\top w_1 = x, \quad \boldsymbol{b}^\top w_m = 0.5x$$

Since $WGI_w = 4\left(\rho - \frac{Q_w^2}{P_w} - \frac{(\rho - Q_w)^2}{1-P_w}\right)$ by equation 5 we get:

$$WGI_w = 4\left(\rho - \frac{(\frac{\boldsymbol{b}^\top \cdot w + \rho}{2})^2}{\frac{\boldsymbol{a}^\top \cdot w + 1}{2}} - \frac{(\rho - \frac{\boldsymbol{b}^\top \cdot w + \rho}{2})^2}{1 - \frac{\boldsymbol{a}^\top \cdot w + 1}{2}}\right) = 4\left(\rho - \frac{(\rho + \boldsymbol{b}^\top \cdot w)^2}{2(1 + \boldsymbol{a}^\top \cdot w)} - \frac{(\rho - \boldsymbol{b}^\top \cdot w)^2}{2(1 - \boldsymbol{a}^\top \cdot w)}\right)$$

We substitute the values of $\rho$, $\boldsymbol{a}^\top \cdot w$ and $\boldsymbol{b}^\top \cdot w$:

$$WGI_{w_0} = 4\left(0.5 - \frac{0.5^2}{2} - \frac{0.5^2}{2}\right) = 1$$

$$WGI_{w_1} = 4\left(0.5 - \frac{(0.5+x)^2}{2} - \frac{(0.5-x)^2}{2}\right) = 4\left(0.25 - x^2\right) = 1 - 4x^2$$

$$WGI_{w_m} = 4\left(0.5 - \frac{(0.5+0.5x)^2}{2} - \frac{(0.5-0.5x)^2}{2}\right) = 1 - x^2$$

Since $w_m = 0.5 \cdot w_0 + 0.5 \cdot w_1$, in order to show non convexity we show that $WGI_{w_m} > 0.5 \cdot WGI_{w_0} + 0.5 \cdot WGI_{w_1}$:

$$1 - x^2 > 0.5 \cdot 1 + 0.5(1 - 4x^2) = 1 - 2x^2$$

which concludes the proof. ■

## Appendix H. Approximate $\gamma$ Margin With a Stochastic Tree

In this appendix we show that the class of decision trees with stochastic linear classifiers has an efficient representation with a bounded error for a $\gamma$ margin feasible linear separator.

**Theorem H.1** *Let $w$ be a $\gamma$ margin feasible linear separator, then there exists a stochastic tree $T$ with feasible linear separators in the internal nodes such that the error of the tree is bounded by $\epsilon$ and the depth of $T$ is $O\left(\ln(\epsilon^{-1})\gamma^{-2}\right)$.*

**Proof** Let $y \in \{0, 1\}$ be a binary label. Let $w$ be a linear separator such that $\forall x : (2y-1) \cdot w \cdot x \geq \gamma$. We denote $Z_i$ as a random variable such that: $\Pr(Z_i = 1) = \frac{1+w\cdot x}{2}$ and $\Pr(Z_i = -1) = \frac{1-w\cdot x}{2}$. Let $S = \sum_{i=0}^{h} Z_i$ be a random variable which is the sum of $h$ independent $Z_i$s. We note that $E[Z_i] = 1 \cdot \Pr(Z_i = 1) - 1 \cdot \Pr(Z_i = -1) = w \cdot x$ and therefore by the expectation of a sum $E[S] = \sum_{i=0}^{h} E[Z_i] = h(w \cdot x)$.

Let $x$ be a positive sample, meaning $w \cdot x \geq \gamma$. We bound the probability of the sum $S$ to be negative by using Hoeffding's inequality:

$$\Pr(S \leq 0) = \Pr(S - E[S] \leq -E[S]) = \Pr(S - E[S] \leq -h(w \cdot x))$$
$$\leq \exp\frac{-2\left(h(w \cdot x)\right)^2}{\sum_{i=0}^{h} 2^2} = \exp\frac{-h\left(w \cdot x\right)^2}{2}$$

Since $w \cdot x \geq \gamma$, we get $\Pr(S \leq 0) \leq \exp\frac{-h\gamma^2}{2}$, and if we want to drive the probability below $\epsilon$:

$$\Pr(S \leq 0) \leq \exp\frac{-h\gamma^2}{2} \leq \epsilon \rightarrow \frac{-h\gamma^2}{2} \leq \ln\epsilon \rightarrow h \geq 2\ln(\epsilon^{-1})\gamma^{-2}$$

Similarly, for a negative sample and $\Pr(S \geq 0) \leq \epsilon$ we also require $h \geq 2\ln(\epsilon^{-1})\gamma^{-2}$.

We now describe a stochastic decision tree $T$: $T$ is a full tree of depth $h \geq 2\ln(\epsilon^{-1})\gamma^{-2}$. Each internal node contains the stochastic linear separator $w$ and each leaf $l \in leaves(T)$ is labeled 1 if the path from $root(T)$ to $l$ contains more left turns than right and 0 otherwise.

We notice that when predicting for sample $x_i$ using $T$, the decision in each internal node is modeled by $Z_i$. A positive sample $x$ is classified positive according to the expectation over the leaves:

$$\Pr(y_i = 1) = \sum_{l \in leaves(T)} label(l) \Pr(Reach(x, l))$$

We consider all the positive labels, since the negative labels contribute 0 to the expectation:

$$\Pr(y_i = 1) = \sum_{l \in leaves(T) | label(l)=1} \Pr(Reach(x, l))$$

By the construction of $T$, this probability equals the probability of reaching leaves with more left turns than right (therefore these leaves are labeled as 1). Namely,

$$\Pr(y_i = 1) = \sum_{l \in leaves(T)} label(l) \Pr(Reach(x, l)) = \Pr(S \geq 0) \geq 1 - \epsilon$$

And therefore the error probability $\Pr(y_i = 0) \leq \epsilon$. Similarly, we can show that the error for a negative sample is also bounded by $\epsilon$. Finally this ensures that the expected error is also smaller than $\epsilon$ as required. ■

## Appendix I. Empirical Evaluations

### I.1. Validating Strong Learnability

In this section we test our classifier in various learning scenarios to verify it captures the target function well. We demonstrate this ability with targets with increasing difficulty as well as unbalanced classes. All experiments were carried in the setting of a 2-dimensional space on random unit size vectors. The reported results are the average of 10 repetitions of each experiment, and the number of internal nodes in the tree is 15 unless otherwise noted.

**Single Hyperplane:** In the first experiment the concept class is a hyperplane which intersects the origin and both classes have equal weight. As figure 8 shows, the algorithm constantly selects hyperplanes which are very close to the target function, and indeed as a result the accuracy of the trained model is around 0.99 which shows that the target is captured well.

**Single Hyperplane and Artificial Bias:** Next, we investigate the behavior when we add a bias term to the classifier. We model the bias term by adding a constant-valued feature to each example. We note that since the decision boundary intersects the origin there is no reason to add this bias term, however, we want to examine the case where the classifier and the target are mismatched (in practice when the target function is unknown it may be beneficial to use a bias in different scenarios).

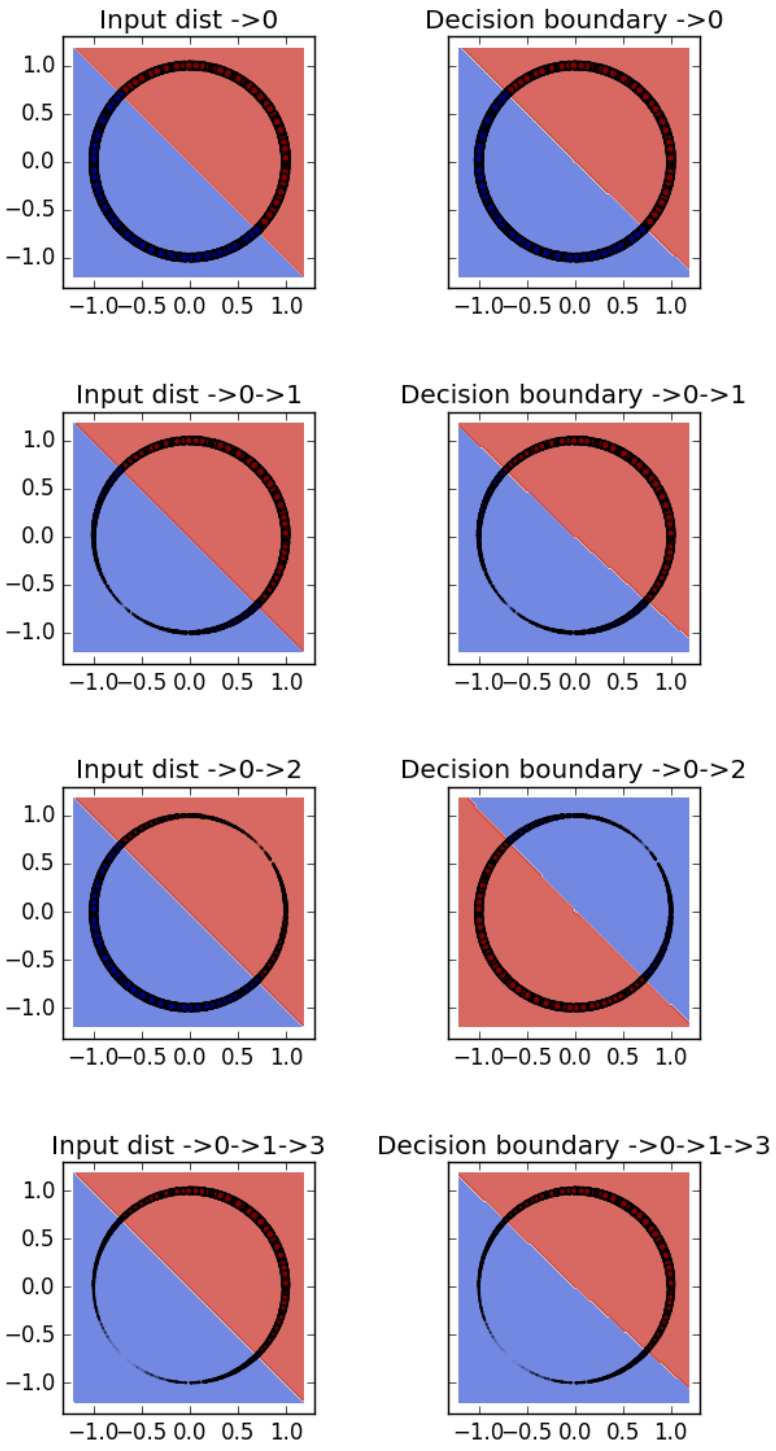


Figure 8: Single Hyperplane Decisions
The decisions taken by the proposed classifier. The title explains the position in the tree hierarchy (root is index 0). On the left the input distribution s.t the circle size is relative to sample weight and the color is based on the true label. On the right the decision of the internal node.

Table 1: Bias effects on confidence

| $b$ bias | $c_b$ positive mean | $\theta_b$ normalization coefficient |
|---|---|---|
| 0 | 0.8 | 1 |
| 1 | 0.71 | 0.707 |
| 1.5 | 0.67 | 0.555 |
| 2 | 0.63 | 0.447 |
| 3 | 0.6 | 0.316 |
| 5 | 0.55 | 0.196 |
| 10 | 0.51 | 0.099 |

Now, the data contains an artificial constant that the classifier should theoretically ignore. A-priori, we know it is possible to cancel the effect of the bias: both $\boldsymbol{a}$ and $\boldsymbol{b}$ will have their last coordinate to be the value of the bias, the decision in each node of the tree - $w$ is a linear combination of the form $w = \alpha\boldsymbol{a} + \beta\boldsymbol{b}$. Therefore, for the bias to be canceled out, the classifier needs to enforce $\alpha - \beta = 0$.

In practice, we indeed see a trend of the classifier to reduce the last coordinate to 0. However as shown in table 1, we also see a drop in the confidence when the bias is increased. This phenomena is explained if we consider the limitations we imposed on the stochastic linear classifier:

When the bias (denoted as $b$) is zero, the data vector $x$ is exactly norm one i.e $1 = \sum x_i^2$. Since the stochastic linear classifier is restricted to consider input vectors of norm $\leq 1$, once a bias $b > 0$ is introduced, all the true data coordinates ($x_i$) are normalized by a factor $\theta_b$ which is:

$$1 = \sum (\theta_b \cdot x_i)^2 + b^2 \rightarrow \theta_b = \frac{1}{\sqrt{1+b^2}}$$

Denote $x_b$ and $w_b$ the data and the classifier if the data is modeled with bias $b$. Even if we assume that $\forall b : w_0 = w_b$ (the algorithm completely ignores the bias) we still need to consider how the classifier output changes because of this added bias. Denote by $c = \frac{w_0 \cdot x_0 + 1}{2}$ (the result of the prediction without bias) thus $w_0 \cdot x_0 = 2c - 1$, the new prediction for bias $b > 0$ is

$$c_b = \frac{w_b \cdot x_b + 1}{2} = \frac{w_0 \cdot x_b + 1}{2} = \frac{\theta_b \cdot w_0 \cdot x_0 + 1}{2} = \theta_b(c - 0.5) + 0.5$$

The first transition is because we assumed that $w_0 = w_b$.

The confidence is the distance $c_b - 0.5$ and therefore for $b$ is $\theta_b(c - 0.5)$ and since $\theta_b < 1$ we have that $c_b$ is always smaller than the confidence for bias 0 (which is $c - 0.5$). In other words, even if we have the same stochastic linear classifier, we lose a factor of $\theta_b$ in confidence due to the restriction $\|x\| \leq 1$. We notice that by weakening the restriction to $\|w \cdot x\| \leq 1$ we could have avoided the reduction in confidence however it is unclear what effect it would have on our solution since the optimization solved in Appendix E.1 will no longer hold.

Table 1 shows the drop in prediction confidence in the first internal node as the bias changes. We note that in our tests the weight in the predictor that matched the bias was reduced to almost 0. The middle column shows that the prediction confidence was indeed reduced according to the relation described above.

**Single Hyperplane with Unbalalnced Labels** In the next test, we show strong learnability even for unbalanced classes. For this experiment we changed the ratio between classes from 1:1 we

had in the previous experiments to 2:1 ratio between the classes. We observed that our classifier's accuracy dropped to 0.94. In order to understand this drop we considered the first split of the tree. We noticed that even though the accuracy dropped, when considering the soft accuracy i.e the average of $\|y_i - \hat{y}_i\|$ we see that both the "basic" balanced case and the unbalanced case both consistently reach to scores around 0.3. We expected the soft accuracy value to be comparatively low compared to the accuracy since we now care about the confidence of the classification and not the sign. We know that the stochastic linear classifier could potentially reach perfect confidence but it is rarely achieved in practice. Therefore the drop in accuracy is attributed to the fact that the classifier tries to optimize the distances to the separating hyperplane and not the sign, but it still preforms well in terms of accuracy (as it still scores accuracy of 0.94).

**Multiple Hyperplanes - XOR** In the next set of experiments the target function is a XOR of the input hyperplanes (XOR is the product of the signs of $w \cdot x$ for every $w$ in the target function). We chose this type of targets, since our classifier is an aggregation of linear decisions and should therefore capture this structure well.

In the first experiment we have two fixed hyperplanes. The reason the hyperplanes are fixed is to make sure the classes are balanced. In the second experiment, we again have two hyperplanes but now these are chosen randomly, which may result in very unbalanced classes (for instance if the two hyperplanes lie very very close to each other). In the last experiment we raise the number of hyperplanes to three.

As expected, the accuracy goes down as the target becomes more complex: in the first experiment the classifier has an accuracy of 0.94 while just changing to unbalanced classes in the second experiment causes the accuracy to drop to 0.9. The third experiment which has a more complex target achieves an initial accuracy of 0.81. For the last experiment we also tested adding more internal splits, raising the number of internal nodes from 15 (which we use as a default parameters) to 30 raises the accuracy to 0.9. Raising the number of nodes to 60 gives just 0.03 more to a total of 0.93. In general we see deceasing accuracy gains as we add more nodes.

Finally, considering all the above experiments we can conclude that the proposed classifier is able to capture these target concepts well (even non linear ones).

### I.2. Comparison to other classifiers

This section discuss empirical evaluations of our classifier on synthetic data-sets we created with the motivation of understanding the benefit of our decision tree using stochastic linear classifiers, compared to decision tree with decision stumps, e.g., CART, and a standard linear classifier, e.g., linear SVM. We selected the simulated data, with the goal of highlighting the similarities and differences between the classifiers. In the following, we described the results of two simulated data sets.

In the first experiment we demonstrate that unlike standard decision tree algorithms, our method fits well a data-set which is labeled according a random linear boundary with noise in **a high dimension**. Specifically, we select a random hyperplane $w$ and each sample $x$ is a random unit vector, where the the label is $y = sign(w \cdot x)$ and dimension is $d = 100$. We add random classification noise, i.e., flip the label with probability of $\theta$. We show the accuracy of our model compared to both CART and linear SVM as a function of the noise rate $\theta$ in Figure 9. Clearly, SVM is ideal for this task, and indeed it outperforms the other two classifiers. We ran both CART and our method to build

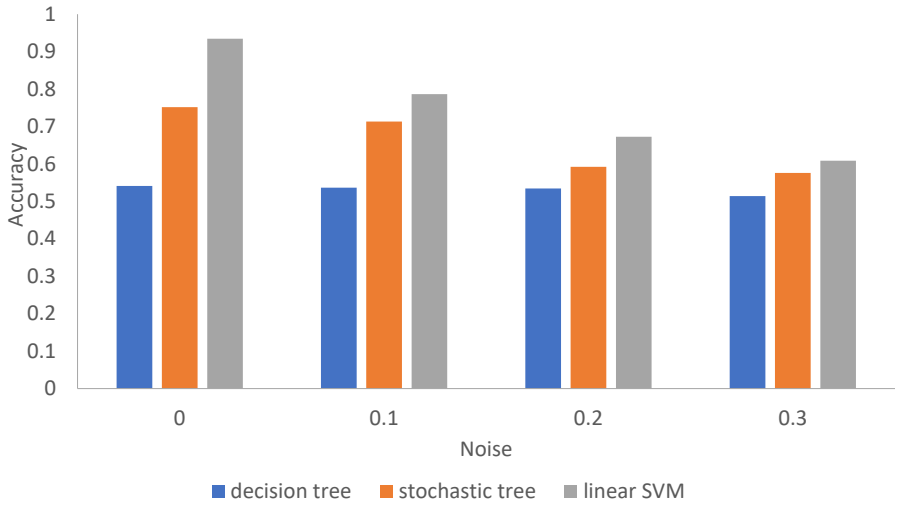


Figure 9: Hyperplane with noise.
Accuracy of the different classifiers for linearly separable points. In orange, our stochastic classifier, blue is CART and grey is linear SVM.

decision trees of depth 10. The results show that our method achieves competitive performance to SVM while outperforming CART.

The second synthetic data set demonstrates the ability of our classifier to handle non-linearly separable examples, where the boundaries are not axis aligned. Again, all samples are random unit vectors of dimension $d$. The labels correspond to the XOR of the samples with two hyperplanes, which are perpendicular to each other and are not axis aligned. Specifically we take $w_1 = [\mathbf{1}_{d/2}, \mathbf{1}_{d/2}]$ and $w_2 = [-\mathbf{1}_{d/2}, \mathbf{1}_{d/2}]$ and the label is $y = sign(x \cdot w_1) \oplus sign(x \cdot w_2)$. Both decision trees were allowed to reach depth of 10 and the sample size is $10,000$. We used various dimension sizes from $d = 6$ to $d = 30$, and the results are plotted in Figure 3.

The results were in line with out intuition. First, Linear SVM achieves the worst performance since it does not have the expressive power required for representing this target function. Second, a decision tree algorithm, such as CART, which considers only a single attribute in each split, requires a significant depth to approximate this high-dimensional XOR. Finally, our method achieves a good accuracy since it is not limited to a single hyperplane, as is Linear SVM, nor is it limited to axis aligned decisions, as is CART. We do note, that since our method is stochastic, we need to continuously repeat a hyperplane in order to make a split more significant. This explains why, due to the limited depth of the decision tree, our performance deteriorate as the dimension increases. An alternative is to limit the number of nodes in the decision tree. Further experiments show that our method can achieve the above performance with only 50 internal nodes (rather than a depth of size 10, which implies 1024 internal nodes).