Predictive Learning Models for Concept Drift

John Case [a], Sanjay Jain [b], Susanne Kaufmann [c], Arun Sharma [d], Frank Stephan [e]

[a] *Department of Computer and Information Sciences, University of Delaware, 101A Smith Hall, DE 19716-2586, USA,* case@cis.udel.edu.

[b] *School of Computing, National University of Singapore, Singapore 119260, Republic of Singapore,* sanjay@comp.nus.edu.sg.

[c] *Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe, 76128 Karlsruhe, Germany, EU,* susanne.kaufmann@t-online.de.

[d] *School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia,* arun@cse.unsw.edu.au.

[e] *Mathematisches Institut, Universität Heidelberg, Im Neuenheimer Feld 294, 69120 Heidelberg, Germany, EU,* fstephan@math.uni-heidelberg.de.

Abstract

Concept drift means that the concept about which data is obtained may shift from time to time, each time after some minimum permanence. Except for this minimum permanence, the concept shifts may not have to satisfy any further requirements and may occur infinitely often. Within this work is studied to what extent it is still possible to predict or learn values for a data sequence produced by drifting concepts. Various ways to measure the quality of such predictions, including martingale betting strategies and density and frequency of correctness, are introduced and compared with one another.

For each of these measures of prediction quality, for some interesting concrete classes, (nearly) optimal bounds on permanence for attaining learnability are established. The concrete classes, from which the drifting concepts are selected, include regular languages accepted by finite automata of bounded size, polynomials of bounded degree, and sequences defined by recurrence relations of bounded size. Some important, restricted cases of drifts are also studied, for example, the case where the intervals of permanence are computable. In the case where the concepts shift only among finitely many possibilities from certain infinite, arguably practical classes, the learning algorithms can be considerably improved.

1 Introduction

In many machine learning situations, the concepts to be learned or the concepts auxiliarily useful to learn may drift with time [2,3,5,6,8,11,19]. As in the just previous references, to sufficiently track drifting concepts to permit learning something of them at all, it is necessary to consider some restrictions on the nature of the drift. For example, Helmbold and Long [8] bound the probability of disagreement between subsequent concepts. Blum and Chalasani [3] place some constraints on how many different concepts may be used, or the frequency of concept switches. Bartlett, Ben-David and Kulkarni [2] consider 'class of legal function sequences' based on some constraints (such as being formed from a walk on a directed graph).

The previous literature on drift considers what can be learned with suitably "slow" or otherwise constrained drift. In some cases lower bounds are shown too. In the general computability setting of the present paper we examine constraints on drift that are absolutely forced (given our criteria of success). Many of our results prove necessary (sometimes surprising) bounds on these constraints in concrete situations. Our upper and lower bounds are often nearly tight in the cases where we supply them. The models we consider are on-line (rather than off-line), and we consider next value extrapolation. In this context we consider several liberal but intrinsically interesting criteria of success (liberal, in part to keep the necessary constraints relatively mild).

More particularly, in the present paper we consider some pleasantly modest, necessary restrictions on the *rate* with which one concept changes into another, model concepts as functions and employ as our principal learning vehicle (computable) martingale betting strategies [9,16]. It is our hope that the present study, with its necessary bounds on natural, concrete cases, may lead to better insight into which kinds of drifting concepts might be successfully tackled in the real world. Also, our use of martingale betting strategies, etc. may suggest to the machine learning practitioner some new approaches for on-line learning in cases where such a style of criteria might be acceptable and workable (and where other criteria might not).

$\mathcal{N}$ denotes the set of natural numbers $\{0, 1, 2, \ldots\}$. Functions (as concepts) considered in this paper have domain $\mathcal{N}$ or, in some special cases, the set of binary strings $\{0, 1\}^*$ which is identified with $\mathcal{N}$ in a standard way. The range of the functions is normally $\mathcal{N}$, but it is sometimes $\{0, 1\}$ (in the case of computable languages represented as characteristic functions) or the set of integers $\mathcal{I}$ or rationals $\mathcal{Q}$ (in the cases of some concrete examples). We sometimes call $\{0, 1\}$-valued functions, *binary functions.*

It is not possible to predict the next values of a rapidly shifting concept if, in each time step, the concept changes without restriction. For example, a drift which randomly vacillates between the constantly 0 function and the constantly 1 function can produce as a data sequence any $\{0, 1\}$-valued function, and, hence, the class of such data sequences cannot be usefully predicted.

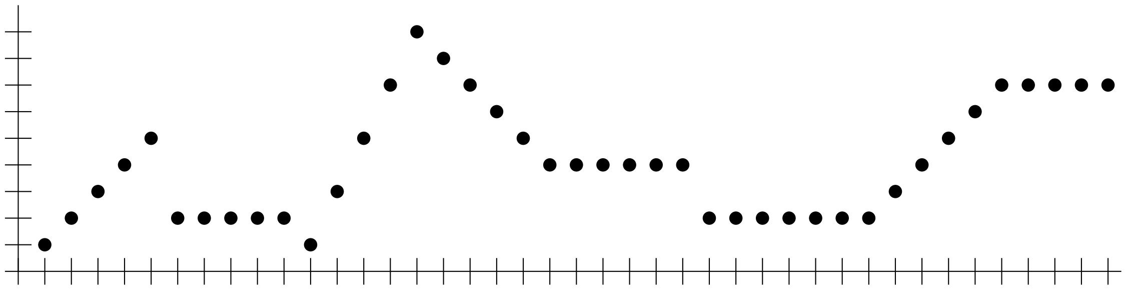
Therefore, given a class $\mathcal{S}$ of functions, the learning tasks we consider involve data sequences for segments of members of $\mathcal{S}$ where these segments do not change from one member of $\mathcal{S}$ to another too often. We require that any concept/function from such an $\mathcal{S}$ in a drifting data sequence be present for some minimal number of successive data points. We call a function p computing this minimal number the *permanence.* The class of data sequences with segments from members of $\mathcal{S}$ with each segment required to be present with permanence p is called $\mathcal{S}[p]$. The formal definition follows immediately.

$|I|$ denotes the length of the interval I.

Definition 1 *Let $\mathcal{S}$ be a class of computable functions. A func-*

tion f is said to be obtained from $\boldsymbol{\mathcal{S}}$ by concept drift with permanence p *if and only if, for each x, there is an interval I_x containing x and a function $g_x \in \boldsymbol{\mathcal{S}}$ such that $|I_x| \geq p(\min(I_x))$ and $f(y) = g_x(y)$, for all $y \in I_x$. $\boldsymbol{\mathcal{S}}[p]$ denotes the class of all such functions f.*

We consider permanence p only such that p is a non-decreasing and $\{1, 2, 3, \ldots\}$-valued function. We always assume such restriction on p without explicitly saying so. If, for example, the basic set S consists of linear functions and the permanence is 5, then every data-item must belong to an interval of length 5 where f follows one linear rule.



Learning deals normally not with a single concept but with a class of concepts. Therefore it is necessary to define when a class of objects is learnable under a given criterion. As we see in the immediately following definition, learnability of a class is defined in terms of learnability of the single objects in it.

Definition 2 *A class $\boldsymbol{\mathcal{S}}$ of functions can be* learned under a given criterion with permanence p *if and only if there is a computable and total machine M which succeeds on every function $f \in \boldsymbol{\mathcal{S}}[p]$ under the given criterion. (Here and below "total" means that the machine always has defined output.)*

So it suffices, then, to define various criteria under which a learner M is said to succeed on a *single* function f. Shortly below we define three such criteria of success.

Learning is normally modeled as a process to identify an underlying global concept which describes the observed behavior. Under concept drift, this underlying global description does not exist or is too complicated. Therefore, the learner can be expected to give local descriptions only. Within this paper, the local behavior is mostly described by just guessing the next value(s). (Since we deal almost always with "learning by prediction" we often just write "M learns f" as a shorthand notation for "M learns f by predicting values of f" and so on.) Because of the unpredictable drifts of the concept, it is unavoidable to err infinitely often. So the learning criteria considered, in effect, involve the ratios of successes and failure during the learning process. The learners studied in the sequel are always total and computable devices which give predictions for the values $f(x+1)$ from the data $f(0), f(1), \ldots, f(x)$. The criteria of correctness for such devices differ in how the quantity of correct and incorrect predictions are measured and compared. The next three definitions introduce learning criteria each of which quantify the amount of correct prediction which is required of a successful learner M operating on a function f (normally in $\boldsymbol{\mathcal{S}}[p]$).

Regarding *frequency identification*, (see Definition 3 just below), note that Kinber and Zeugmann [10] previously studied an interesting *off-line* (hence, different) criteria of frequency learning or identification.

Definition 3 *A learner M* learns *a function f (or* predicts f*)* with frequency a out of b *if and only if, for each x, at least a of the equations*

$$f(y+1) = M(f(0)f(1)\ldots f(y))$$

are correct, where y ranges over the b arguments $x, x+1, \ldots, x+b-1$. We refer to such learners as frequency learners*.*

We say that a class is frequency learnable *if and only if some learner predicts all functions in the class with frequency a out of b, for some a, b, with $1 \leq a \leq b$.*

The requirement that, for each interval of length b, at least a of the predictions are correct is quite restrictive. This could be alleviated somewhat by aiming for a particular ratio between a and b in a limiting sense instead of requiring it for each interval. In other words, the set X of all correct predictions need only be of some *minimum* "density." We employ a notion of density introduced by Tennenbaum [14, §9.5/9-38] in formalizing this approach to frequency learners. Tennenbaum called the limit inferior* of the sequence $\frac{1}{x+1} \cdot (A(0) + A(1) + \ldots + A(x))$ the *density* of the set A.** Royer [15] introduced the related notion of *uniform density* of a set A to be the limit inferior of the sequence $\min\{\frac{1}{x+1} \cdot (A(y) + A(y+1) + \ldots + A(y+x)) : y \in \boldsymbol{\mathcal{N}}\}$. These notions are incorporated in the next definition.

Definition 4 *A learner M* learns *a function f (or* predicts *f)* with (uniform) density *q if and only if the (uniform) density of the set $\{x : M(f(0)f(1)\ldots f(x)) = f(x+1)\}$ is at least q. We refer to such learners as* (uniform) density learners.

It may be argued that in the criteria introduced so far, the learner is unnecessarily penalized by being required to make a prediction at all times. The learner is not allowed to use any knowledge about the times when predictions are easy and when they are difficult. The learner may be bogged down by difficult predictions even if it has some restricted knowledge which is enough to correctly predict the majority of values. A well-known setting that models such a case is the world of gambling [9]. Here a gambler may decide whether and how much to bet on a certain prediction coming true or whether to pass if it is too difficult to make a prediction with a reasonable chance of success. Such a gambling learner is said to succeed if and only if it can extract enough information about the values of f so that successive bet-

* The definition of the limit inferior can be found in most advanced calculus text books, for example, [18]. The limit inferior of a sequence $a_0, a_1, \ldots$, is the supremum r of all rational numbers q which are below almost all a_n: $r = \text{supremum}\{q : (\forall^\infty n)\,[q < a_n]\}$

**For $A \subseteq \boldsymbol{\mathcal{N}}$, $A(x) = 1$ if $x \in A$ and $A(x) = 0$ if $x \notin A$.

ting (predicting) allows it to accumulate arbitrarily large amount of money. The following definition introduces this criterion via *martingales.*

Definition 5 *A* martingale *is a computable function m from strings to positive rational numbers such that, for every string σ, there is a symbol a and a rational q such that*

- $0 \leq q < m(\sigma)$;
- $m(\sigma a) = m(\sigma) + q$ *and* $m(\sigma b) = m(\sigma) - q$, *for* $b \neq a$.

The martingale m learns *a function f (or* succeeds on *f or* wins on *f) if and only if the function $x \rightarrow m(f(0)f(1)\ldots f(x))$ is not bounded by any constant.*

Intuitively, the martingale calculates the accumulated wealth of a player who, for every sequence or string σ, bets an amount of money q that (a number) a will follow σ and receives it in the case of success and loses it otherwise. This definition includes the ability to pass by betting 0 and also the ability to bet arbitrary small amounts of money. That is, there is no smallest unit like a "Cent" which cannot be split into smaller pieces. On the other hand, the player cannot (in our definition) go broke by playing at some time his total accumulation at that time. This latter constraint is for expository convenience in the present paper — we avoid having to test for going broke — and our results hold with or without it.

A martingale *wins* iff — according to the previous example — the gambler has arbitrary large amounts of money at some suitable time. This analogy becomes more striking by the fact, that the definition of martingale learning is invariant under the following change of definition.

> A martingale m *learns* f iff the limit inferior of $m(f(0)f(1)...f(x))$ is ∞, that is, iff, for all c, for all but finitely many x,
> $m(f(0)f(1)...f(x)) > c$.

This is interesting since, when successful, the money of the gambler exceeds any given bound c *almost always* and *not only infinitely often.*

Any of the above criteria requires that the learner correctly predicts infinitely often on functions to be learned. One might say that this is an essential precondition for any kind of learning process. Hence we call a learning criterion *reasonable*, if it explicitly as above or at least implicitly requires that the learner M predicts each function to be learned infinitely often correctly. The class of all binary functions is not learnable with respect to a reasonable criterion: if M is a learner then one constructs a binary function f inductively by $f(0) = 0$; $f(x+1) = 1$, if $M(f(0)f(1)\dots f(x)) \downarrow = 0$, and $f(x+1) = 0$, otherwise. This function f disagrees with every prediction of M. So any criterion which allows to learn the class of all the binary functions is not reasonable. Frequency learning, martingale learning, and learning with a density $q > 0$ are reasonable criteria; learning with density 0 is not reasonable since the requirement for success is void.

In the sequel we proceed as follows.

In Section 2, we compare the relative predictive ability of martingale learners, frequency learners and density learners. We show that frequency learners are the most restrictive, while martingale learners and density learners with low density (below $\frac{1}{2}$) are incomparable generalizations of them.

In Section 3, we analyze the learnability of several interesting concrete concept classes under the various criteria introduced in the present section. Our upper bounds on permanence are also shown to be (nearly) optimal.

We show that, for all $h \in \boldsymbol{\mathcal{N}} - \{0\}$, if constant permanence p satisfies $p > (3h+3)\log(h+3)$, then $\boldsymbol{\mathcal{S}}[p]$ is frequency learnable, where $\boldsymbol{\mathcal{S}}$ is the class of the regular languages over the alphabet $\{0, 1\}$ accepted by finite automata with up to h states. Note that here and in the subsequent sections, logarithms are base 2.

While polynomials of bounded degree are shown to be learnable with reasonable constant permanence under all our criteria, we show that the natural concept class of pattern languages [1] *with erasing* separates martingale learners from density learners (also from frequency learners and uniform density learners). A martingale learner succeeds on the erasing pattern languages already at the surprising small constant permanence 7.

Fibonacci and other sequences defined by similar recurrence relations grow exponentially, yet we show such classes defined by bounded size of recurrence relations are learnable with reasonable constant permanence under all our criteria.

While Sections 2 and 3 deal with drifts having no restrictions except for permanence bounds, Section 4 is devoted to some natural restrictions on drift like (a) the resulting function has to be computable, (b) the set $\boldsymbol{\mathcal{N}}$ is *computably* partitioned into disjoint intervals $I_0, I_1, \ldots$ such that each I_n has at least $p(\min(I_n))$ elements and each $f \in \boldsymbol{\mathcal{S}}[p]$ presented to the learner agrees on each interval I_n with some function $g_n \in \boldsymbol{\mathcal{S}}$ and (c) the drift vacillates between a finite number of functions in $\boldsymbol{\mathcal{S}}$. In each case, it is shown that there are classes $\boldsymbol{\mathcal{S}}$ and permanences p such that the class $\boldsymbol{\mathcal{S}}[p]'$ consisting of all functions $f \in \boldsymbol{\mathcal{S}}[p]$ satisfying an additional restriction on the drift can be learned with some smaller permanence or sharper learning criterion than the class $\boldsymbol{\mathcal{S}}[p]$. Hence, there are always situations where that restriction on the drift pays off, that is, where knowledge of some regularity within the drift allows construction of *better* learning algorithms.

Any computability terminology used below and not explained herein may be found in [14].

The first result states that everything that can be learned by a frequency learner can also be learned by a martingale learner. The strategy employed by the martingale learner is the well known doubling-algorithm which sometimes ruins gamblers but which nicely works in this case.

Proposition 6 *Suppose a class, $\boldsymbol{\mathcal{S}}$, of functions (possibly but not necessarily generated by some concept drift) is frequency learnable. Then $\boldsymbol{\mathcal{S}}$ can be learned by a martingale.*

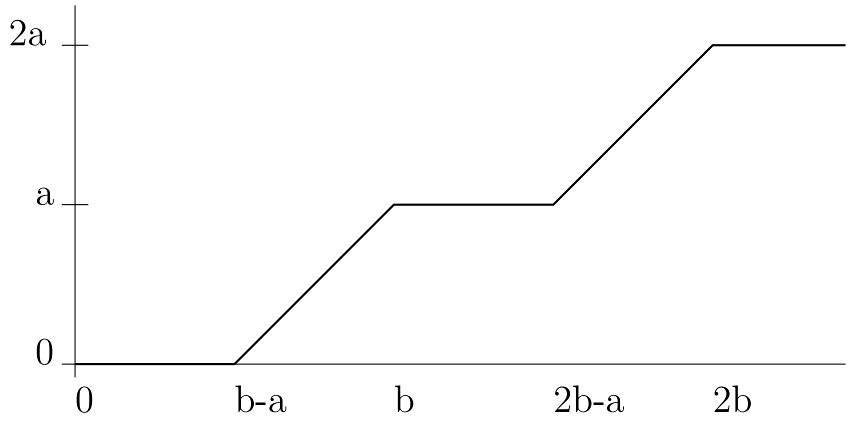
Proof. Suppose M predicts a class of functions with frequency 1 out of b. Note that M makes infinitely many correct predictions and never a sequence of b consecutive wrong predictions.

Now a martingale m is constructed which wins on all functions on which M succeeds with frequency 1 out of b. This is done by just making the same predictions as M, but using the ability to choose the amount of money to bet in such a way that, at each correct prediction, losses since the last correct prediction are compensated.

In the beginning, the initial capital is divided into 2^b units. The martingale m bets money as follows. After ℓ consecutive failures since the last success (or since the beginning), it bets 2^ℓ units. Note that, due to the doubling of amount bet, each success wins 1 more unit than the loss incurred since the last success. Moreover, since there can be at most $b-1$ consecutive losses, the martingale never goes bankrupt. It is easy to verify that the amount of money that the martingale has after the k-th successful prediction is $2^b + k$. So the strategy pays off in the limit. ∎

The next result investigates the inclusion relation on frequency learning for different parameters. We first introduce some definitions. In the following the natural numbers a, b, c, d always satisfy $1 \leq a \leq b$ and $1 \leq c \leq d$. Let $F_{a,b}(bx + y) = ax$, for

$y = 0, 1, \ldots, b-a$, and $F_{a,b}(bx+y) = ax+y$, for $y = -a+1, \ldots, 0$. Note that, for every natural number d, it is possible to find $x \in \boldsymbol{\mathcal{N}}$, $y \in \boldsymbol{\mathcal{I}}$ with $-a < y \leq b-a$, such that $d = bx + y$. For all a, b and d it holds that $\frac{ad}{b} - a < F_{a,b}(d) \leq \frac{ad}{b}$. The graph of $F_{a,b}$ looks like this:



Theorem 7 *Every class learnable with frequency a out of b is also learnable with frequency c out of d if $c \leq F_{a,b}(d)$. If $c > F_{a,b}(d)$, then there exists a class of functions which is learnable with frequency a out of b, but not with frequency c out of d.*

Proof. For the first part, let $c \leq F_{a,b}(d)$ and suppose M predicts $\boldsymbol{\mathcal{S}}$ with frequency a out of b. We claim that M also predicts $\boldsymbol{\mathcal{S}}$ with frequency c out of d. Let f be an arbitrary function in $\boldsymbol{\mathcal{S}}$. Suppose $d = bx + y$, where $-a < y \leq b - a$. The proof now proceeds based on whether y is positive.

(a): $y \geq 0$. Then $F_{a,b}(d) = ax$. Since M predicts correctly a values of f on every interval of length b, M also predicts correctly ax values of f on every interval of length bx which can be viewed upon as a union of x disjoint intervals. Since $d \geq bx$ it follows that M predicts at least $ax = F_{a,b}(d)$ values correctly on an interval of length d.

(b): $y < 0$. Then $F_{a,b}(d) = ax + y$. For the ease of notation let $z = -y$ and $d = bx - z$, z is positive. Again one knows that M predicts correctly ax values of f on an interval of length bx. From these predictions at most z can be correct on the last z arguments. So M makes at least $F_{a,b}(d) = ax - z$ correct predictions on any interval of length $d = bx - z$.

So in both cases M predicts f correctly on each interval of length d at least $F_{a,b}(d)$ times, in particular at least c times. So M learns f with frequency c out of d.

For the second part of the theorem, consider the class of all primitive recursive functions which take the value 0 on the set $X = \{xb + y : y \in \{0, 1, \ldots, a-1\}\}$. $\boldsymbol{\mathcal{S}}[p]$ is then the set of all functions (also the noncomputable ones) which are 0 on the set X. For every learner M, there is a function $f \in \boldsymbol{\mathcal{S}}[p]$ which differs from the predicted value on every $z \notin X$. So starting with any input of the form $z = xb + a - 1$, M correctly predicts $f(z+u)$, for $u = 1, 2, \ldots, d$, only if $z + u$ is in X. Thus the number of correct predictions is at most $|X \cap \{xb + a, xb + a + 1, \ldots, xb + a + d - 1\}| = F_{a,b}(d)$. This completes the proof. ▮

Fact 8 *For the notion of predicting with density and uniform density the following results hold.*
(a) If $\boldsymbol{\mathcal{S}}[p]$ is learnable with uniform density q, then $\boldsymbol{\mathcal{S}}[p]$ is also learnable with density q. On the other hand, there is a class $\boldsymbol{\mathcal{S}}$ such that, for every permanence p, $\boldsymbol{\mathcal{S}}[p]$ is learnable with density 1 but not with any uniform density $q > 0$.
(b) If $\boldsymbol{\mathcal{S}}[p]$ is learnable with frequency a out of b, then $\boldsymbol{\mathcal{S}}[p]$ is also learnable with uniform density $\frac{a}{b}$.
(c) Some $\boldsymbol{\mathcal{S}}[p]$ is learnable with density $\frac{1}{2}$ but not by any martingale.
(d) If $\boldsymbol{\mathcal{S}}[p]$ is learnable with density $q > \frac{1}{2}$, then it is also learnable by a martingale.

Proof. (a): This implication of learning with uniform density towards learning with normal density follows directly from the definition. The separation follows ideas of Royer [15]. Consider the class $\boldsymbol{\mathcal{S}}$ of all primitive recursive functions which are 0 on the set $X = \{x : (\exists y)\,[2^y < x < 2^{y+1} - y]\,\}$. Then $\boldsymbol{\mathcal{S}}[p]$ contains all total functions which are 0 on X. So an algorithm which predicts 0 everywhere is correct on X. On the other hand, for any M, there is an $f \in \boldsymbol{\mathcal{S}}[p]$ which differs from the predictions on every input outside X. So $\boldsymbol{\mathcal{S}}[p]$ can be learned with density 1 (since X has density 1) but not with positive uniform density (since X has the uniform density 0).

(b): Let M be any learner which predicts all $f \in \boldsymbol{\mathcal{S}}[p]$ with frequency a out of b. Then, for any interval of length d, M predicts all $f \in \boldsymbol{\mathcal{S}}[p]$ with frequency $F_{a,b}(d)$ out of d. Since $F_{a,b}(d) > \frac{ad}{b} - a$, it follows that M learns $\boldsymbol{\mathcal{S}}[p]$ with uniform density $\lim_{d\to\infty} \frac{1}{d} \cdot F_{a,b}(d) = \frac{a}{b}$.

(c): Let $p(x) = 2x$. Let $\boldsymbol{\mathcal{S}}$ be the class of all primitive recursive $\{0,1\}$-valued functions g which satisfy $\frac{x+1}{2} - 2\log(x) \leq g(0) + g(1) + \ldots + g(x) \leq \frac{x+1}{2} + 2\log(x)$, for all $x \geq 1$. There is a random function f which also satisfies this relation for all $x \geq 1$ [13]. This f is in $\boldsymbol{\mathcal{S}}[p]$, for any permanence p, since every prefix of f is extended by some $g \in \boldsymbol{\mathcal{S}}$. On the other hand, this sequence is not learnable by a martingale because of its randomness.

So it remains to show that just predicting 1 gives correctness density $\frac{1}{2}$ or more. Fix $f \in \boldsymbol{\mathcal{S}}[p]$. We show that the sequence $x \rightarrow \frac{f(0)+f(1)+\ldots+f(x)}{x+1}$ has the limit inferior $\frac{1}{2}$. Let the intervals I_x be as in Definition 1. If both, x and y, belong to the same interval I_z then $f(x+1)+f(x+2)+\ldots+f(y) \geq \frac{y-x}{2} - 2\log(x) - 2\log(y)$. We define a sequence $x_0, x_1, \ldots$ as follows. $x_0 = 1$ and x_{n+1} be the least number $x' > x_n$ such that either $x' \geq 2x_n$ or no $I_{x''}$ contains both x_n and $x'+1$. Note that f agrees with some $g \in \boldsymbol{\mathcal{S}}$ on the arguments $x_n+1, x_n+2, \ldots, x_{n+1}$. Then $x_{n+2} \geq 2x_n$, since either (I) $x_{n+1} \geq 2x_n$ or (II) $x_{n+2} \geq 2x_{n+1}$ or (III) $x_n, x_{n+2} \notin I_{x_{n+1}}$. The conditions (I) and (II) clearly imply $x_{n+2} \geq 2x_n$, using the fact that $x_{n+2} \geq x_{n+1} \geq x_n$; the condition (III) implies $x_{n+2} \geq 2x_n$ using the facts that $x_n < \min(I_{x_{n+1}})$, $x_{n+2} > \max(I_{x_{n+1}})$ and $\max(I_{x_{n+1}}) \geq 2\min(I_{x_{n+1}})$. Thus, for any n and any x between x_n and x_{n+1}, one has that $f(0)+f(1)+\ldots+f(x) \geq \frac{x-x_n}{2} + \frac{x_n - x_{n-1}}{2} + \ldots + \frac{x_1 - x_0}{2} - 2\log(x) - 4\log(x_n) - 4\log(x_{n-1}) - 4\log(x_1)$

$- 2\log(x_0) \geq \frac{x}{2} - 4(n+1)\log(x)$. Since $x_{m+2} \geq 2x_m$, for all m, one has that $x_n \geq 2^{n/2}$ and $n \leq 2\log(x_n)$. It follows that the sum $f(0) + f(1) + \ldots + f(x)$ is greater than $\frac{x}{2} - 12(\log(x))^2$ and the sequence $x \to \frac{f(0)+f(1)+\ldots+f(x)}{x+1}$ has the limit inferior (and also the limit superior) $\frac{1}{2}$.

(d): Schnorr [16, Section 10] shows that every binary function not learnable by a martingale satisfies the law of large numbers, that is, the density of 1's converges to $\frac{1}{2}$. Furthermore he showed that if the density of 1's is larger than $\frac{1}{2}$, then some martingale succeeds by always betting a suitable amount of money on 1. Similarly one can argue, for $\boldsymbol{\mathcal{S}}[p]$ learnable by M with density $q > 1/2$, that some martingale succeeds on $\boldsymbol{\mathcal{S}}[p]$, by betting always a suitable amount of money on the value predicted by M (since these predictions are correct on a set of density $q > \frac{1}{2}$). ▮

The results of Fact 8 have some straightforward extensions: Learnability by martingales can also be obtained if $\boldsymbol{\mathcal{S}}$ contains only functions f which are learnable via some fixed machine under some uniform density $q_f > 0$ — or, equivalently, which are learnable under some frequency 1 out of b_f. That means, that it is more important that all functions in the given class are learnable by the same learner than that they are learnable with respect to the same parameters. The other way, to fix the parameter but not the machine, does not help since every computable function is predictable with frequency 1 out of 1 — by its own program — but the class of all computable functions is not learnable by a martingale [16].

3 Concrete Classes

In this section optimal and nearly optimal bounds are derived for the permanence necessary and sufficient to learn certain concrete classes under drift.

Suppose $\boldsymbol{\mathcal{S}}$ is a class of up to k binary functions. We first investigate for which constant (depending on k) permanence p the class $\boldsymbol{\mathcal{S}}[p]$ is frequency learnable.

Looking at the class of all binary functions which repeat with period $\lfloor \log(k) \rfloor$ one sees directly that the condition $p > \log(k)$ is *necessary* — otherwise the class $\boldsymbol{\mathcal{S}}[p]$ contains every binary function and is not learnable under every reasonable criterion. On the other hand, there is an upper bound that is only a bit above this lower bound. The problem which gives an upper bound slightly larger than the expected value $\lfloor \log(k) + 1 \rfloor$, is that one does not explicitly know the intervals on which f coincides with some g from the concept class. So the learner intuitively has to assume that these intervals may be chosen by an adversary. The implicit bound on p in the next theorem could also be made a bit more explicit by taking stronger sufficient conditions such as $p \geq \log(k) + 2\log\log(k+1) + 10$ or $p \geq \log(k) + \log\log(k+1) + 2\log\log\log(k+3) + 10$.

Theorem 9 *Suppose $\boldsymbol{\mathcal{S}}$ contains up to k computable $\{0,1\}$-valued functions and nothing else. Then $\boldsymbol{\mathcal{S}}[p]$ is frequency learnable if $p - \log(p) > \log(k)$.*

Proof. Fix k and corresponding p. Since all functions in $\boldsymbol{\mathcal{S}}$ are computable and permanence is constant, it is possible to compute on any interval $x+1, x+2, \ldots, x+b$, the finite set F_x of all possible value-vectors $(f(x+1), f(x+2), \ldots, f(x+b))$, where f ranges over $\boldsymbol{\mathcal{S}}[p]$. Whenever there is a constant b such that $|F_x| < 2^b$, for all x, then one can predict one of the values in the given interval by the well-known halving algorithm. By restarting this process after any successful prediction one can show that $\boldsymbol{\mathcal{S}}[p]$ is predictable with frequency 1 out of b. So it remains to find such a b.

Let $b = 2p-1$. Any interval I' of length b contains a subinterval I of length p on which f equals some $g \in \boldsymbol{\mathcal{S}}$. The behavior of f on I' can be described by the starting point of I, which is among the first p positions of the interval, the index of the function $g \in \boldsymbol{\mathcal{S}}$, which coincides with f on I, and $p-1$ binary bits to represent the remaining values of f. Thus, there are $k \cdot p \cdot 2^{p-1}$ possibilities which f can take on the interval I'. Since $\log(k) < p - \log(p)$, we have $\log(k) + \log(p) < p$, $k \cdot p < 2^p$ and $k \cdot p \cdot 2^{p-1} < 2^b$ which is the desired combinatorial condition. ▮

As an application of this theorem, the class of all regular languages accepted by some deterministic finite automaton having at most h states, can be learned in the presence of concept drift with *constant* permanence (where, of course, the constant depends on h).

Example 10 *Suppose $\boldsymbol{\mathcal{S}}$ is the class of the regular languages over the alphabet $\{0,1\}$ accepted by deterministic finite automata with up to h states. Then, $\boldsymbol{\mathcal{S}}[p]$ is frequency learnable, if $p - \log(p) > 3h\log(h+1)$. (For example, $p - \log(p) > 3h\log(h+1)$ holds if $p \geq (3h+3)\log(h+3)$.)*

For $p \leq h$, $\boldsymbol{\mathcal{S}}[p]$ is not learnable under any reasonable learning criterion, since $\boldsymbol{\mathcal{S}}[p]$ is the class of all the binary functions.

Proof. The positive result is obtained by first noting that there are at most $(2h^2)^h$ deterministic finite automata with h states — those with fewer states are also covered since they just might have additional inaccessible states. Each state can be accepting or rejecting which gives the term 2^h within the product. For each state the transition for input 0 and for input 1 can access one of the other h states. This gives the term h^{2h} in the product above. Letting $k = (2h^2)^h$ one uses the upper bound $3h\log(h+1) \geq h(2\log(h) + \log(2)) = h\log(2h^2)$ for $\log(k)$ to obtain the sufficient condition from Theorem 9.

We now consider the second part of the theorem. Suppose $p = h$. Suppose, a binary string $a_0a_1\ldots a_n$ represents the binary number $(1a_0a_1\ldots a_n) - 1$. Any binary function, f, on the natural numbers which is periodic, with period h, can be represented with a finite state automaton of h states as follows.

For $y < h$, state y of the automaton is accepting iff $f(xh+y) = 1$. The automaton starts in state 0 (which represents the empty string). Transition from state v to state w is done on input a, iff $w = 2v + a + 1$ modulo h. Note that, on any input binary string s, the automaton ends up in state y iff the natural number corresponding to string s is of the form $xh + y$.

For each interval, $xh, xh + 1, \ldots, xh + h - 1$, one can find a periodic function g interpolating any given binary values on this interval. Thus every binary function is in $\boldsymbol{\mathcal{S}}[p]$. Therefore $\boldsymbol{\mathcal{S}}[p]$ is not learnable according to any reasonable learning criterion. ∎

Finding the best permanence often requires considerable combinatorics. Some classes, such as polynomials, are easier to handle where a full solution of the possible learning frequencies in dependence of the allowed degree and permanence is possible. The proof of Theorem 11 (c) furthermore gives the more general result that a class, which contains an extension of every function with finite domain, is not learnable under concept drift. The same principle holds if only the binary functions with finite domain are extended. Thus, one can obtain another proof for the second statement in the previous example.

Theorem 11 *Let k be a natural number and $\boldsymbol{\mathcal{S}}$ be the class of all polynomials of degree up to k.*
(a) $\boldsymbol{\mathcal{S}}[k+1]$ contains every function and thus $\boldsymbol{\mathcal{S}}$ cannot be learned with permanence $k+1$ under any reasonable learning criterion.
(b) If $h > k+1$, then $\boldsymbol{\mathcal{S}}[h]$ is learnable with frequency a out of b iff $a \leq F_{h-k-1,h}(b)$.
(c) Let $\boldsymbol{\mathcal{S}}$ be the class of all polynomials. Then, for every permanence p, the class $\boldsymbol{\mathcal{S}}[p]$ contains all total functions and thus is not learnable under any reasonable criterion.

Proof. (a): Let $I_n = \{n(k+1), n(k+1)+1, \ldots, n(k+1)+k\}$; the intervals I_0, I_1, … form a partition of $\boldsymbol{\mathcal{N}}$ and each interval contains exactly $k+1$ elements. Given any function f, one can find for each n a polynomial g_n of degree up to k which is equal to f on I_n. Thus,

$$(\forall f)\,(\forall n)\,(\exists g_n \in \boldsymbol{\mathcal{S}})\,(\forall x \in I_n)\,[g_n(x) = f(x)]$$

and $\boldsymbol{\mathcal{S}}[k+1]$ contains all the total functions.

(b): For the positive result, with $a \leq F_{h-k-1,h}(b)$, it is sufficient to show that $\boldsymbol{\mathcal{S}}$ can be frequency learned with frequency $h-k-1$ out of h. The learner M predicts 0 for $f(0), f(1), \ldots, f(k)$ and M predicts $g_x(x+k+1)$ for $f(x+k+1)$, where g_x is the polynomial of least degree which coincides with f on $f(x)$, $f(x+1)$, …, $f(x+k)$. Let $I = \{y, y+1, \ldots, y+h-1\}$ be an interval of length h and assume that $y+u$ is the first place where the prediction algorithm makes an error. $y+u$ must belong to some interval J of length h on which f coincides with some polynomial g of degree up to k. Since M errs, u must be among the first $k+1$ elements of J. So M makes at least $h-k-1$ correct predictions on the input $y+u$, $y+u+1$, …, $y+u+h-1$. Since M makes in total $u+h-k-1$ correct predictions on the interval $\{y, y+1, \ldots, y+u+h-1\}$, it follows that M makes at least $h-k-1$ correct predictions on the interval $\{y, y+1, \ldots, y+h-1\}$.

Next we consider the converse direction. Given any learner M, one can use the intervals $I_n = \{hn, hn+1, \ldots, hn+h-1\}$ and find, for each n, a polynomial g_n of degree not above k, such that $g_n(hn+u) = M(f(0)f(1)\ldots f(hn+u-1)) + 1$, for $u = 0, 1, \ldots, k$. Let $f = g_n$ on I_n. This inductive procedure gives a function f such that M fails to predict $f(x)$ correctly, whenever x is in $\{0, 1, \ldots, k\}$ modulo h. It follows that, if M learns f with frequency a out of b, then $a \leq F_{h-k-1,h}(b)$ must hold.

(c): This is similar to case (a). The growing permanence is compensated by the absence of any degree bound. Choosing a partition $I_0, I_1, \ldots$ of $\boldsymbol{\mathcal{N}}$, respecting the permanence, one can find, for each function f and each natural number n, a polynomial g_n, which agrees with f on I_n. Thus, $\boldsymbol{\mathcal{S}}[p]$ contains every total function. ▮

The values of polynomials can be computed from the preceding ones. So a linear function satisfies the equation $f(x+2) = 2f(x+1) - f(x)$ and a quadratic function satisfies $f(x+3) = 3f(x+2) - 3f(x+1) + f(x)$. The functions satisfying such equations are a natural generalization of polynomials. The Fibonacci numbers, given by $f(x+2) = f(x) + f(x+1)$, and the powers of 2, given by $f(x+1) = 2f(x)$, cannot be represented by polynomials and demonstrate that the generalization is proper. In the case of polynomials, it was necessary to bound the degree in order to achieve learnability. For the generalization, this bound is given by the number of terms on the right-hand side of the recurrence relation (1).

Example 12 *Let $\mathcal{S}$ be the class of functions defined by a finite recurrence relation*

$$f(x+k+1) = a_0 f(x) + a_1 f(x+1) + \ldots + a_k f(x+k), \quad (1)$$

where the values $f(0)$, $f(1)$, …, $f(k)$ can be chosen arbitrarily. This class $\mathcal{S}$ is frequency learnable with permanence $2k+3$ but not *with permanence $k+2$.*

Proof. Let V be the set of all possible parameters $(a_0, a_1, \ldots, a_k)$. V is a vector set of dimension $k+1$. For each tuple $(a_0, a_1, \ldots, a_k)$ one calls

$$\begin{aligned}\max\{y \le x : (\forall z)\, [\text{ if } x - y \le z < x \text{ then} \\ f(z+k+1) = a_0 f(z) + a_1 f(z+1) + \ldots + a_k f(z+k)\,]\}\end{aligned}$$

the confidence of this tuple of parameters at x. The confidence is always at least 0 and at most x. In order to predict $f(x+k+1)$ one computes a tuple $(a_0, a_1, \ldots, a_k)$ with maximal confidence at x — if there are several possibilities it does not matter which one is taken — and outputs the value $a_0 f(x) + a_1 f(x+1) + \ldots + a_k f(x+k)$ for this tuple.

For verification one works with the superclass of all $\boldsymbol{\mathcal{Q}}$-valued functions which is of course more difficult to learn than the integer-valued functions satisfying (1). Next one shows that at least one of the predictions for $f(x+k+1)$, $f(x+k+2)$, …, $f(x+2k+2)$ is correct if the concept is the same at x, $x+1$, …, $x+2k+2$. Let V_h denote the class of all recurrence relations consistent with the input data on the interval from x to $x+k+h$. Each set V_h is a nonempty vector set containing a relation valid for the function f on the data between x and $x+k+h$. Predicting f at $x+k+h+1$ always uses a tuple of parameters from V_h. So each wrong prediction causes at least one tuple being removed from V_{h+1} and therefore reduces the dimension of V_{h+1} (in comparison to V_h) by at least 1. Since V_0 has dimension $k+1$ and V_{k+2} has dimension at least 0, it follows that there are at most $k+1$ wrong predictions.

So whenever the permanence is at least $2k+3$, within any interval of length at least $4k+5$ some concept is correct for a subinterval $\{x, x+1, \ldots, x+2k+2\}$ of length $2k+3$. Thus at least one of the $k+2$ values $f(x+k+1)$, $f(x+k+2)$, …, $f(2k+2)$ is predicted correctly. Thus the class $\boldsymbol{\mathcal{S}}$ is learnable with frequency 1 out of $4k+5$.

The lower bound $k+2$ is due to periodic functions of the form $f(x+k+1) = f(x)$ or the form $f(x+k+1) = -f(x)$ which can interpolate any sequence in $\{-1,1\}^{k+2}$ on intervals of length $k+2$. ▮

It is quite natural to ask whether the lower bound can be lifted to $2k+2$. The following example illustrates that a lower bound $2k+2$ would need some nontrivial properties of the space of the values which perhaps are present in the field $\boldsymbol{\mathcal{Q}}$ and in the ring $\boldsymbol{\mathcal{I}}$ of the integers but which are certainly not present in the Boolean field $\{0,1\}$.

Example 13 *Let $\boldsymbol{\mathcal{S}}$ be the class of functions defined by a finite* *recurrence relation*

$$g(x+k+1) = a_0 g(x) + a_1 g(x+1) + \ldots + a_k g(x+k),$$

over the Boolean field $\{0,1\}$ *where the multiplication is the "Boolean and", the addition is the "Boolean exclusive or" and the values* $g(0)$, $g(1)$, $\ldots$, $g(k)$ *can be chosen arbitrarily. This class* $\boldsymbol{\mathcal{S}}$ *is frequency learnable with permanence* $2k+2$.

Proof. The learning algorithm is easy in this case. It just says 0 to predict $f(x(k+2))$ and says 1 to predict $f(x(k+2)+y)$ for $y = 1, 2, \ldots, k+1$. We now show that, if $x(k+2) > 2^{k+1} + k$, then at least one of the predictions for f on the interval from $x(k+2) - k - 1$ to $x(k+2) + k + 1$ is correct. Thus the whole class $\boldsymbol{\mathcal{S}}[2k+2]$ is learnable with frequency 1 out of $2^{k+1} + 2k + 2$ and density $\frac{1}{2k+3}$.

This algorithm is based on some special properties which are outlined now. First, each function $g \in \boldsymbol{\mathcal{S}}$ is periodic from 2^{k+1} on, with a period whose length is also at most 2^{k+1}: there are only 2^{k+1} many different vectors $(g(x), g(x+1), \ldots, g(x+k))$ and thus two of them must be equal, say $(g(x), g(x+1), \ldots, g(x+k)) = (g(x+y), g(x+y+1), \ldots, g(x+y+k))$, where $0 \leq x < x+y \leq 2^{k+1}$. It follows that $g(z+y) = g(z)$ for all $z \geq x$. Second, if g has $k+1$ consecutive 0's then it remains at 0 thereafter (since, if $g(x) = g(x+1) = \cdots = g(x+k) = 0$, then by induction, for all $r \geq 1$, $g(x+k+r) = a_0 0 + a_1 0 + \ldots + a_k 0 = 0$) and thus takes 0 at any place beyond 2^{k+1} because of the periodicity.

Given any number of the form $x(k+2) > 2^{k+1}$ the values of some f belong to the same concept g on some interval of length $2k+2$ containing $x(k+2)$. If now $g(x(k+2)) \neq 0$ then g is not 0 on $k+1$ consecutive places and thus takes 1 somewhere on the interval from $x(k+2) - k - 1$ to $x(k+2) - 1$ and on the interval from $x(k+2) + 1$ to $x(k+2) + k + 1$. Hence, either the first half of the interval where f and g coincide is below $x(k+2)$, or the second half of this interval is beyond $x(k+2)$. Therefore, f takes a 1 somewhere on the interval from $x(k+2) - k - 1$ to $x(k+2) - 1$

or somewhere on the interval from $x(k+2)+1$ to $x(k+2)+k+1$. So one of the predictions on the interval from $x(k+2)-k-1$ to $x(k+2)+k+1$ is correct. ∎

The *pattern languages* [1] are a prominent and natural language class. We consider a known natural extension with the aim of showing that some natural class $\boldsymbol{\mathcal{S}}$ separates the ability to learn by a martingale from that to learn by a frequency learner.

We employ a *dyadic* coding of the Boolean strings 1-1, onto the natural numbers and identify each such string with its natural number code. In this coding the empty string represents the natural number 0 and the length one Boolean strings 0, 1 represent the natural numbers 1, 2, respectively. For an arbitrary Boolean string $b_{i_k}b_{i_{k-1}}\ldots b_{i_1}b_{i_0}$, we write d_{i_j} for the element of $\{1,2\}$ coding the Boolean bit b_{i_j} and we code $b_{i_k}b_{i_{k-1}}\ldots b_{i_1}b_{i_0}$ by the natural number $d_{i_k}2^k+d_{i_{k-1}}2^{k-1}+\ldots+d_{i_1}2^1+d_{i_0}2^0$. Hence, for example, the code of the Boolean string 00 is 3, and that of the string 111 is 14. A *pattern* is a string consisting of variables and (Boolean) constants. It generates the language of all words which can be obtained by replacing each variable by a binary string. A pattern language [1] is called *erasing* if the variables in the defining pattern may be replaced by the empty string. So the pattern $0x1xy$ generates words like 01, 010, 011, 0010, 00100, 00101 and so on, but it does not generate the words 0000 and 11111 since the constants 0 and 1 cannot be removed. We refer the reader to the paper [17] for a nice survey on results about pattern languages.

Example 14 *If $\boldsymbol{\mathcal{S}}$ is the class of all erasing pattern languages then $\boldsymbol{\mathcal{S}}[7]$ can be learned by a martingale but $\boldsymbol{\mathcal{S}}[p]$ is not frequency learnable even for very fast growing permanences p. For constant permanence p, it is also impossible to learn $\boldsymbol{\mathcal{S}}[p]$ with some density $q>0$.*

Proof. λ denotes the empty string.

The first result is based on the fact that every pattern language containing the words 0^n00, 0^n01 and 0^n11 also contains 0^n10. To see this, note, that the pattern has to end with some variables since otherwise either 0^n00 or 0^n01 are not in the language. Furthermore the variable generating the 1 in 0^n01 occurs only once. If one takes the substitution λ for all other variables, we get that, for some m, the language generated by 0^mx is contained in the language generated by the given pattern. In particular $m \leq n$ (since otherwise the pattern could not generate 0^n11) and thus 0^n10 is in the language by taking $x = 0^{n-2-m}10$. An analysis similar to above also holds with 0 and 1 interchanged.

Considering $\boldsymbol{\mathcal{S}}[7]$, one observes that a martingale can use the fact that no function $f \in \boldsymbol{\mathcal{S}}[7]$ takes the characteristic function $1, 0, 1, 1, 1, 1, 0, 1$ on the strings 1^n00, 1^n01, 1^n10, 1^n11, $0^{n+1}00$, $0^{n+1}01$, $0^{n+1}10$, $0^{n+1}11$ (since either the first four or the last four strings are evaluated by the same pattern. Thus whenever 1^n00, 1^n10 and 1^n11 in the first case or $0^{n+1}00$, $0^{n+1}01$ and $0^{n+1}11$ in the second case belong to the language represented by f, so is also 1^n01 or $0^{n+1}10$, respectively). So one can correctly predict, for each n, the function f on one of the eight strings 1^n00, 1^n01, 1^n10, 1^n11, $0^{n+1}00$, $0^{n+1}01$, $0^{n+1}10$, $0^{n+1}11$. Following the basic idea of the proof of Proposition 6, a martingale can translate this knowledge into a winning strategy for functions in $\boldsymbol{\mathcal{S}}[p]$.

The second result that $\boldsymbol{\mathcal{S}}[p]$ is not frequency learnable for any p can be obtained by showing that there are infinitely many places where — provided that the concept is drifting at these places — it is impossible to correctly predict any of the next 2^n inputs. That is, for any given predictor, one constructs an $f \in \boldsymbol{\mathcal{S}}[p]$ such that there are infinitely many strings z_n for which $f(z_nu)$ differs from the predicted value for all $u \in \{0,1\}^n$. The drift takes place just before the places z_n0^n and the z_n can be made so large that every interval between z_n0^n and $z_{n+1}0^{n+1}$ has the minimum length required by the permanence.

Let $z_n = 0110\,0^{a(n)n}1^{a(n)n}$ — where $n \to a(n)$ is a function necessary to satisfy a rapidly growing permanence and in addition

$a(n) \geq 2$ for all n. Fix a predictor M. For each n and each $u \in \{0,1\}^n$, let $f(z_n u) = 0$ iff M predicts 1 for $f(z_n u)$ and let $f(z_n u) = 1$ otherwise. Note that M predicts wrongly on $z_n\{0,1\}^n$, for each n. We will define below a pattern which interpolates the values of f on $z_n\{0,1\}^n$; the values between $z_n 0^n$ up to the last string before $z_{n+1}0^n$ will follow the so defined n-th pattern. Let $\{u_0, u_1, \ldots, u_h\} = \{u \in \{0,1\}^n : f(z_n u) = 1\}$ be the set of the suffixes u for which $f(z_n u) = 1$.

The pattern which interpolates f on all inputs in $z_n\{0,1\}^n$ is

$$x_0y_0x_1y_1 \ldots x_ny_n\, y_0x_0y_1x_1 \ldots y_nx_n 0^{a(n)n}1^{a(n)n}\sigma_0(u_0)\sigma_1(u_1)\ldots\sigma_h(u_h)$$

where $\sigma_m(u_m)$ is a string of length n, consisting only of the variables x_m and y_m, such that the i-th variable in the string is x_m, if $u_m[i] = 0$ and y_m, if $u_m[i] = 1$. So $\sigma_m(01101) = x_my_my_mx_my_m$. Then any word $0110\,0^{a(n)n}1^{a(n)n}\,u_m$ is generated by the pattern (by taking $x_m = 0$, $y_m = 1$ and all other variables to be λ).

On the other hand, due to length constraints the constant part of any word of length $4 + n + 2a(n)n$ generated by the pattern has the constant part at the same middle position. So the part generated by $x_0y_0x_1y_1 \ldots x_ny_n\, y_0x_0y_1x_1 \ldots y_nx_n$ is 0110.

There are at most three ways to generate this part. (I): One variable generates at least two characters. In this case all other variables are λ and the prefix has the form x_mx_m which cannot generate 0110. So this case does not arise. (II): x_m and y_m both generate one character each. Then $x_m = 0$ and $y_m = 1$ and the generated word is $0110\,0^{a(n)n}1^{a(n)n}\,u_m$. (III): Two variables, say x_i and y_j, both generate one character each, where $i \neq j$ since the equality matches the previous case (II). Then these two variables occur in the sequence $x_iy_jx_iy_j$ or $y_jx_iy_jx_i$, none of which can generate the prefix 0110. The same holds for the cases when these variables are x_i and x_j or y_i and y_j, where $i \neq j$. So this case also does not arise.

Putting the cases (I), (II) and (III) together it follows that f coincides with the language generated by the pattern on the arguments $z_n\{0,1\}^n$. So it is not possible to learn $\boldsymbol{\mathcal{S}}[p]$ with frequency 1 out of 2^n, for all n. That is, the erasing pattern languages are not frequency learnable under any concept drift.

The third result is that $\boldsymbol{\mathcal{S}}[p]$ is not learnable with positive (uniform) density for constant p. In the case of uniform density, this is already covered by the proof of the second result. For nonuniform density, the proof must be adapted in such a way that the predictions do not only fail completely on some intervals, but fail completely on arbitrary long intervals.

There are two modifications compared to the previous proof. First the permanence is constant. Thus one can choose a different suitable pattern on almost all intervals of the form $z\{0,1\}^n$, with $4n \leq |z| < 4n+5$. Second it is shown that a suitable pattern can interpolate a given function f, if the Kolmogorov complexity of z is sufficiently high. So one can take a diagonalizing f such that, for z, x with $4n \leq |z| < 4n+5$ and $|x| = n$, $f(zx)$ is 0 if z is too small or the Kolmogorov complexity of z is too small; and f diagonalizes a given learner M whenever z is long enough and its Kolmogorov complexity is sufficiently high. So let n be so large that $2^n > p$ and the Kolmogorov complexity [13] of any string of the form $0^*1^*0^*1^*0^*1^*$ of length up to $4n+4$ is below $3n-1$.

Let z be any string of Kolmogorov complexity above $3n$, satisfying the length constraint $4n \leq |z| \leq 4n+4$ and having the form $z = 0^a01v10w$ (or $1^a10v01w$ which can be dealt symmetrically) where 10 is first occurrence of this form after the prefix 0^a01; therefore $v \in 0^*1^*$. The pattern

$$0^a\, x_0y_0x_1y_1 \ldots x_hy_h\, v\, y_0x_0y_1x_1 \ldots y_hx_h\, w\, \sigma_0(u_0)\sigma_1(u_1) \ldots \sigma_h(u_h)$$

interpolates f on $z\{0,1\}^n$, where $u_m \in \{0,1\}^n$ are the strings satisfying $f(zu_m) = 1$ and h is the number of such strings (here $\sigma_m(u_m)$ depends on u_m as above). Next consider any substitution of the above pattern which generates a string of length $|z|+n$. In this case, one can show that if $x_0y_0x_1y_1 \ldots x_hy_h \neq 01$ and

$y_0x_0y_1x_1 \dots y_hx_h \neq 10$, then one can compute z from 0^a, the two strings $x_0y_0x_1y_1 \dots x_hy_h$ and $y_0x_0y_1x_1 \dots y_hx_h$, the string v and perhaps the last four characters of w. a can be expressed by an expression of size $\log(a) \leq \log(4n)$. v is of the form 0^b1^c and thus can be expressed by an expression of size $2\log(4n+4)$. The last 4 characters of w need an expression of size 4 to be described. The strings $x_0y_0x_1y_1 \dots x_hy_h$ and $y_0x_0y_1x_1 \dots y_hx_h$ can be described using $n+4$ bits, since the constant parts 0^a, v and w already cover $|z|-4$ bits. So the total string z could be described using a constant plus $n+2\log(n)$ bits, which contradicts the fact that z has Kolmogorov complexity at least $3n$, for sufficiently large n.

Thus, if p is constant, $q > 0$ and n is large enough, then, for all intervals of the form $z\{0,1\}^n$, where $4n \leq |z| \leq 4n+4$ and $K(z) \geq 3n$, one might find concepts which contradict all predicted values on these intervals. Every sufficiently long string z' with $K(z') \geq \frac{5|z'|}{6}$ can be represented in such a form. Thus for every given learner M and every constant permanence p, there is an $f \in \boldsymbol{\mathcal{S}}[p]$ such that M fails to predict f on almost all strings z' with Kolmogorov complexity $K(z') \geq \frac{5|z'|}{6}$. These strings have density 1 among the set of all strings. Thus M does not learn f with any density $q > 0$. ∎

4 Restrictions on Drift

The previous section dealt with arbitrary drift and therefore the learning algorithms intuitively had to compensate for drifts produced by an arbitrarily unpleasant adversary. One might argue that nature does not always follow the worst case but is sometimes more pleasant and well-behaved. In particular, drifting concepts might follow some rules and laws; the next three subsections are devoted to discussing the influence of such rules on the ability to learn under concept drift. So we derive conditions under which the subclass $\boldsymbol{\mathcal{S}}[p]' \subseteq \boldsymbol{\mathcal{S}}[p]$ of the functions resulting from a particular restricted drift may be (and are) easier to learn. As will be seen, the exact meaning of the notation $\mathcal{S}[p]'$ will change from one subsection of this section to the next.

4.1 Drifts Preserving Computability

Let REC be the set of all total computable functions. The present section investigates the case where the drift results in computable functions, that is, where $\mathcal{S}[p]' = \mathcal{S}[p] \cap \text{REC}$. The results of Sections 2 and 3 carry over to the case where $\mathcal{S}[p]'$ is used instead of $\mathcal{S}[p]$; provided that in the places where something is "not learnable under any reasonable learning criterion", this statement is weakened to "not learnable under any criterion which does not permit the learnability of all binary recursive functions." It is quite obvious that the inclusions in the previous results go through. However, the noninclusions requires some additional work: instead of taking an arbitrary function for diagonalization one has to construct, for every computable learner, a specific computable function in $\mathcal{S}[p]$ on which this learner fails. The next quite easy example shows how to do this.

Example 15 *There is a class $\mathcal{S}$ and permanence p such that $\mathcal{S}[p]$ is frequency 2 out of 3 learnable but not frequency 3 out of 4 learnable even under computable concept drift.*

Proof. Let $\mathcal{S}$ be the class of all constant functions and consider the class $\mathcal{S}[3]'$. It is easy to see that the algorithm always predicting the last value received so far succeeds to show that $\mathcal{S}[3]$ is learnable with frequency 2 out of 3. Next assume that M is an arbitrary frequency learner and consider the function f given by $f(0) = 0$, and, for $x \in \mathcal{N}$ and $y \in \{1, 2, 3\}$,

$$f(3x+y) = \begin{cases} 0 & \text{if } y = 3 \text{ and } M(f(0)f(1)\dots f(3x+2)) > 0; \\ 1 & \text{if } y = 3 \text{ and } M(f(0)f(1)\dots f(3x+2)) = 0; \\ f(3x) & \text{if } y = 1 \text{ or } y = 2. \end{cases}$$

Since M is computable and total, so is f. Furthermore M makes a prediction error on every number of the form $3x+3$. In particular, if one looks at the predictions at $3x+3$, $3x+4$, $3x+5$, $3x+6$, then at most two of them are correct. Therefore there is no computable machine which learns the class of constant functions with frequency 3 out of 4 and permanence 3. ▮

Some criteria like learning with a fixed frequency a out of b either succeed on a function f or fail already on some finite prefix of f. So whenever such a learner fails on some f one can abstain from changing the concept after this failure. So, if a given learner fails on some $f \in \boldsymbol{\mathcal{S}}[p]$, then it also fails on some computable f from the same class. Thus the question whether $\boldsymbol{\mathcal{S}}[p]$ is learnable with frequency a out of b does not depend on the decision whether all or only the computable functions in $\boldsymbol{\mathcal{S}}[p]$ have to be learned.

However, for the other learning criteria, at best it can only be known in the limit whether the learner is successful or not. So, for certain problems, one can compensate early errors by a lot of good predictions. For these criteria it can be an essential difference whether the learner has to cope with the whole class $\boldsymbol{\mathcal{S}}[p]$ or only the subclass $\boldsymbol{\mathcal{S}}[p]'$ of all computable functions in $\boldsymbol{\mathcal{S}}[p]$. In particular the next theorem shows that there are classes where this transition allows a large improvement in learnability.

Theorem 16 *There is a class $\boldsymbol{\mathcal{S}}$ of computable functions such that, for any p, the class $\boldsymbol{\mathcal{S}}[p]$ cannot be learned under any reasonable learning criterion. However, the subclass $\boldsymbol{\mathcal{S}}[p]' \subseteq \boldsymbol{\mathcal{S}}[p]$ is learnable with uniform density* 1.

Proof. Let A be an immune set [14, §8.2] and define $\boldsymbol{\mathcal{S}}$ as:

$$g \in \boldsymbol{\mathcal{S}} \Leftrightarrow$$

$$g \text{ is computable and } (\forall x)\,[g(x) = 0 \vee g(x) \in A - \{1, 2, \ldots, x\}.]$$

For every predictor M, there is a function $f \in \boldsymbol{\mathcal{S}}[p]$, such that M fails to predict f: $f(x+1)$ is just the first $y \in A$ with $y > M(f(0)f(1)\ldots f(x)) + x$. For any p, f is in $\boldsymbol{\mathcal{S}}[p]$, since it coincides on each interval $\{x, x+1, \ldots, p(x)\}$ with the function g given by

$g(y) = f(y)$, for $y \leq p(x)$, and $g(y) = 0$, for $y > p(x)$. It is easy to verify that $g \in \boldsymbol{\mathcal{S}}$.

On the other hand, $\boldsymbol{\mathcal{S}}[p]'$ contains only computable functions. Since A is immune and since the range of any $f \in \boldsymbol{\mathcal{S}}[p]'$ is a subset of $A \cup \{0\}$, it follows that the range of f is finite and has some maximum, say x. For $y > x$, $f(y)$ must coincide with $g(y)$, for some $g \in \boldsymbol{\mathcal{S}}$; from $g(y) < y$ it then follows that $g(y) = 0$ and therefore also $f(y) = 0$. So $\boldsymbol{\mathcal{S}}[p]'$ contains only functions which are almost everywhere 0. Thus $\boldsymbol{\mathcal{S}}[p]'$ can be predicted with (uniform) density 1. ▮

4.2 Permanence on Disjoint Computable Intervals

The second model limits the drift by requiring *computable* intervals $I_0, I_1, I_2, \ldots$ partitioning $\boldsymbol{\mathcal{N}}$ on which the function to be predicted equals some concept in $\boldsymbol{\mathcal{S}}$; we use $\boldsymbol{\mathcal{S}}[p]'$ to denote the drift class formed in this fashion, where $I_0, I_1, I_2, \ldots$ is understood. The next two examples deal with the case where the learner has — as some kind of additional information — a program computing the intervals $I_0, I_1, I_2, \ldots$ while the last example deals with learners ignorant of the actual intervals. Those learners then only exploit the fact that the intervals are disjoint and therefore also work if the intervals are a non-computable partitioning of $\boldsymbol{\mathcal{N}}$.

Example 17 *If $\boldsymbol{\mathcal{S}}$ contains up to k finite functions and $p > \log(k)$, then the functions in $\boldsymbol{\mathcal{S}}[p]$ respecting the computable intervals $I_0, I_1, \ldots$, are frequency learnable by just using the majority vote algorithm on each interval I_n. The frequency is 1 out of $2\lfloor \log(k) \rfloor + 1$.*

One might argue that such an improvement is due only to the ease of finding an algorithm and not to any real difference between the two concepts. The next example shows that there is a class $\boldsymbol{\mathcal{S}}$ such that $\boldsymbol{\mathcal{S}}'[2]$ is frequency learnable for computable intervals while the general class $\boldsymbol{\mathcal{S}}[2]$ is not learnable under any reasonable criterion for arbitrary drift.

Example 18 *Let $\boldsymbol{\mathcal{S}}$ be the class of all increasing binary functions, that is, $\boldsymbol{\mathcal{S}} = \{0^n 1^\infty : n \in \boldsymbol{\mathcal{N}}\}$. Let $I_0, I_1, \ldots$ be a computable partition of $\boldsymbol{\mathcal{N}}$ such that every interval I_n contains at least two elements. Let $\boldsymbol{\mathcal{S}}[2]'$ be the class of all functions $f \in \boldsymbol{\mathcal{S}}[2]$ which in addition coincide with some $g_n \in \boldsymbol{\mathcal{S}}$ on every interval I_n. Then, $\boldsymbol{\mathcal{S}}[2]'$ is frequency learnable while $\boldsymbol{\mathcal{S}}[2]$ itself is not learnable under any reasonable learning criterion.*

Proof. The learner for $\boldsymbol{\mathcal{S}}[2]'$ predicts $f(x) = 0$, if $x = \min(I_n)$ for some n, and predicts $f(x-1)$ otherwise. So the learner makes a prediction error only if $x = \min\{y \in I_n : f(y) = 1\}$ for some n, that is, there is at most one error per interval. Since the length of the intervals is at least 2, there are never more than two consecutive errors, which occur in the adversary case, at the end of the last and the beginning of the new interval. So $\boldsymbol{\mathcal{S}}[2]'$ is learnable with frequency 1 out of 3.

For the second result on the nonlearnability of the whole class $\boldsymbol{\mathcal{S}}[2]$, consider any $\{0, 1\}$-valued function f with $f(0) = 0$. If it vacillates infinitely often between 0 and 1, then one can split $\boldsymbol{\mathcal{N}}$ into finite intervals I_n such that f takes on I_n first some 0's and then some 1's; for example, if $f \succeq 01001100010$, then $I_0 = \{0, 1\}$ where f takes 01, $I_1 = \{2, 3, 4, 5\}$ where f takes 0011 and $I_2 = \{6, 7, 8, 9\}$ where f takes 0001. Each of these intervals has length at least 2 and f is increasing on these intervals. If f converges to 0 or 1, then one splits $\boldsymbol{\mathcal{N}}$ into initial intervals as above and then has a rest I' on which f either equals 0^∞ or $0^n 1^\infty$. This rest I' can then be divided into intervals of length 2 on which f is also nondecreasing. Thus $\boldsymbol{\mathcal{S}}[2]$ contains all $\{0, 1\}$-valued functions with $f(0) = 0$ and is thus not learnable. ▮

The construction above can be made effective in the following sense: If M is a total recursive machine, then, from an index of M, one can construct a recursive function f not learned by M in the sense that every prediction at some place $x > 0$ is wrong and also construct a computable sequence of intervals $I_0, I_1, I_2, \ldots$ partitioning $\mathcal{N}$ such that f coincides with some function 0^n1^∞ on every interval. So the positive part is mainly due to the fact, that the learner knows the intervals $I_0, I_1, I_2, \ldots$ and without this knowledge, no successful learning is possible.

The next example shows that disjointness itself without knowing the intervals can already yield advantages in terms of learnability with higher frequencies. In contrast to the previous examples, one has one single machine which, for every partition, learns the functions in the *corresponding* class $\mathcal{S}[2]'$ with frequency 2 out of 5. This learner succeeds even for nonrecursive partitions and does not need any a priori knowledge on the actual positions of the intervals.

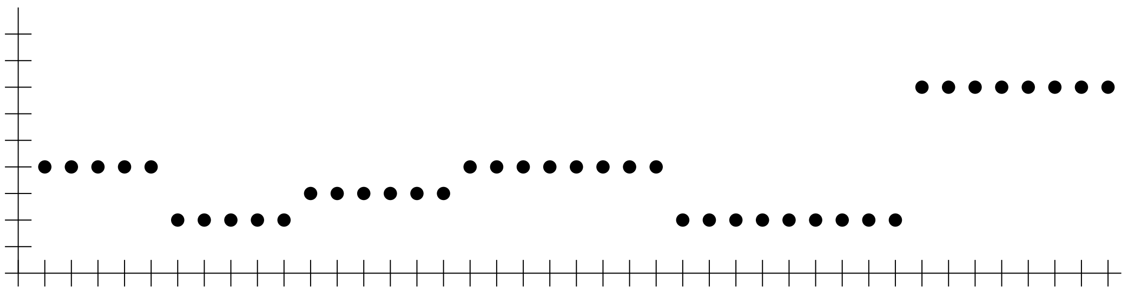
Example 19 *Let $\mathcal{S}$ contain all functions which are 0 at all but one argument and let $I_0, I_1, I_2, \ldots$ be a (not necessarily recursive) partition of $\mathcal{N}$ into intervals of length at least 2. Then the subclass $\mathcal{S}[2]'$ of all functions $f \in \mathcal{S}[2]$ which coincide with functions in $\mathcal{S}$ on the intervals $I_0, I_1, I_2, \ldots$ is learnable with frequency 2 out of 5. The whole class $\mathcal{S}[2]$ is not learnable with frequency 2 out of 5, though it is learnable with frequency 2 out of 6. The corresponding densities of the best possible learning algorithms are $\frac{1}{2}$ and $\frac{1}{3}$.*

Proof. Consider any $f \in \mathcal{S}[2]'$. For any n, there is a $g_n \in \mathcal{S}$ with $f = g_n$ on I_n. Consider an algorithm that always predicts 0. Let J be an interval of length 5. J intersects at most three of the intervals $I_0, I_1, \ldots$ and thus, $f(x) \neq 0$ for at most three $x \in J$. Therefore the above algorithm, which always predicts 0, is correct on 2 of the arguments in J. Thus, $\mathcal{S}[2]'$ is learnable with frequency 2 out of 5 by this algorithm. Furthermore, the learning algorithm predicts correctly with density $\frac{1}{2}$, since, for any n, f is 0 on at least half of the inputs from $I_1 \cup I_2 \cup \ldots \cup I_n$.

For the general case $\mathcal{S}[2]$ and any interval $J = \{x, x+1, x+2\}$ of length 3 one knows that f must coincide with some $g \in \mathcal{S}$ either on $\{x, x+1\}$ or $\{x+1, x+2\}$. So if $f(x+1) \neq 0$ then either $f(x) = 0$ or $f(x+2) = 0$. Therefore the algorithm always predicting 0 is again a frequency learner for $\boldsymbol{\mathcal{S}}[2]$ — but with the reduced rate 1 out of 3. The class $\boldsymbol{\mathcal{S}}[p]$ is also learnable with frequency 2 out of 6 and — by Fact 8 (b) — $\boldsymbol{\mathcal{S}}[p]$ is also learnable with (uniform) density $\frac{1}{3}$. On the other hand one should note that every f, with $f(3x+1) = 0$, for all x, is in $\boldsymbol{\mathcal{S}}[2]$: one can take intervals as $\{3x, 3x{+}1\}$ and $\{3x{+}1, 3x{+}2\}$. So, for each predictor M, there is an f which differs from the predictions for all inputs of the form $3x+2$ and $3x+3$. Thus $\boldsymbol{\mathcal{S}}[p]$ is not learnable with any density $q > \frac{1}{3}$ and — by using the contrapositive of Fact 8 — not learnable with frequency 2 out of 5. ▮

4.3 Vacillating Drift

There are cases where, in principle, a drifting concept might involve any members of some infinite class but, in reality, the drift is only between finitely many of them, for example, between finitely many constant functions.



In this case, this knowledge can be exploited to achieve real improvements in learnability.

As in the case of computable drift, vacillation cannot be exploited for frequency learners. However an improvement can be observed for other types of learning considered in this paper, that is, for martingale learners, learners with some density and learners with uniform density.

It should be noted that such an improvement is possible on many practical classes and not only on some artificially constructed examples as in the case of computable drift. These examples are the class of all polynomials for the case of constant permanence and any uniformly enumerable class for the case of nonconstant permanence.

Example 20 *Let p be constant and $\boldsymbol{\mathcal{S}}[p]'$ denote the class of all functions which vacillate between a finite number of polynomials each with permanence p. Then $\boldsymbol{\mathcal{S}}[p]'$ can be learned with uniform density $\frac{p-1}{p}$.*

Proof. Let $g_0, g_1, \ldots$ be a 1–1 enumeration of all the polynomials. Now the learner M searches on input $f(0)f(1)\ldots f(x)$ for the first k such that $g_k(x) = f(x)$. Then M outputs $g_k(x+1)$ as a prediction for the next value:

$$M(f(0)f(1)\ldots f(x)) = g_{\min\{k : g_k(x) = f(x)\}}(x+1).$$

For the verification of this algorithm, fix $f \in \boldsymbol{\mathcal{S}}[p]'$. There is an h such that f vacillates only between the functions $g_0, g_1, \ldots, g_h$. Any two distinct polynomials agree on only finitely many arguments. So there is a y such that all the polynomials $g_0, g_1, \ldots, g_h$ are different at each $x \geq y$.

Let $x \geq y$ and assume that the prediction for $f(x+1)$ fails. There is a unique $k \leq h$ with $f(x+1) = g_k(x+1)$. By assumption $f(x) \neq g_k(x)$ since the prediction failed. Then f and g_k coincide at an interval of length p containing $x+1$ and not x. Thus the predictions for f at $x+2, x+3, \ldots, x+p$ are correct. Hence, each wrong prediction is followed by at least $p-1$ correct ones. It follows that M, on an interval of length x, makes at most $y + \frac{x}{p}$ mistakes. Thus M learns this f and also all other functions in $\boldsymbol{\mathcal{S}}[p]'$ with uniform density $\frac{p-1}{p}$. ∎

The above algorithm works for the special case of polynomials and there is no directly general equivalent. For example, if $\boldsymbol{\mathcal{S}}$ is the class of all periodic functions, then no learner achieves some minimum density on all functions in $\boldsymbol{\mathcal{S}}[p]'$ for constant permanence p. Hereby a function is *periodic* iff there is a y such that $f(x+y) = f(x)$ for all x. However, in the case of unbounded permanence, that is, in the case that p is not decreasing and not bounded by any constant, it is possible to learn the class $\boldsymbol{\mathcal{S}}[p]'$ with uniform density 1.

Theorem 21 *Let $\boldsymbol{\mathcal{S}} = g_0, g_1, \ldots$ be an effectively enumerable class of total functions and let p be a computable non-decreasing and unbounded permanence. Then the class $\boldsymbol{\mathcal{S}}[p]'$ of all functions in $\boldsymbol{\mathcal{S}}[p]$ which vacillate between finitely many functions in $\boldsymbol{\mathcal{S}}$ can be learned with uniform density* 1.

Proof. The algorithm M for the general problem follows the same basic idea as the one for the polynomials, but needs some more explicit conditions and bounds since certain beautiful properties are absent. Without loss of generality one might assume that any finite function is extended by some g_k; this can be achieved by joining an enumeration of all polynomials to the originally given one. The condition whether:

> for each y with $y+p(y) \leq x$, there is an $h \leq k$, and an interval I of length $p(\min(I))$ containing y such that f agrees with g_h on I,

can be checked effectively in x and k. Such a number k is called a legal bound for f (at the data $f(0), f(1), \ldots, f(x)$). Such a legal bound is always found since there is always some g_k coinciding with f on $0, 1, \ldots, x$. Furthermore, this legal bound converges in the limit (since $f \in \boldsymbol{\mathcal{S}}[p]$ is obtained by vacillating among finitely many functions in $\boldsymbol{\mathcal{S}}$). The basic idea of the learner M is to remember the index of the function (in $g_0, g_1, \ldots$) used for the last prediction, and, in the case the last prediction was wrong, to use cyclically the next index from the set $\{0, 1, \ldots, k\}$ of currently legal indices. For $f(0)$ the algorithm predicts $g_0(0)$ and for $f(x+1)$ it works as follows.

On input $f(0), f(1), \ldots, f(x)$,

M computes a legal bound k for these data.

Suppose h' is the index used for the last prediction $g_{h'}(x)$. Let

$$h = \begin{cases} h' & \text{if } g_{h'}(x) = f(x); \\ h'+1 & \text{if } g_{h'}(x) \neq f(x) \text{ and } h' < k; \\ 0 & \text{otherwise, that is, } g_{h'}(x) \neq f(x) \text{ and } h' \geq k. \end{cases}$$

Then M predicts $g_h(x+1)$.

One can argue that, for large enough x, the above algorithm uses a fixed constant k. If I is an interval on which f coincides with some g_h, $h \leq k$, then M makes at most k false predictions on I: M goes cyclically through the indices $0, 1, \ldots, k$ and since each prediction error induces a cyclic change of the hypothesis, M reaches the correct hypothesis after at most k errors and then does not make any further error until reaching the end of I. Since p is unbounded, for each real $r > 0$, there is an x such that every interval of length x can be covered by at most $r \cdot x$ intervals I on which f coincides with some concept g_h, $h \leq k$. Thus, except for finitely many errors (say c) due to M not having reached the limiting value of the legal bound k, the number of further errors by M on intervals of length x is at most rxk. So if one chooses, for given $q < 1$, an r and corresponding x such that $rxk+c \leq (1-q)x$, then M outputs on each interval of length x at least qx correct predictions. Thus, M learns $\boldsymbol{\mathcal{S}}[p]'$ with uniform density q, for every $q < 1$. It follows that M learns $\boldsymbol{\mathcal{S}}[p]'$ with uniform density 1. ▮

5 Density Learning of Predictions Versus of Programs

The notions of density learners and uniform density learners presented above in the present paper are different from those introduced by Fulk and Jain [7, Definitions 6, 7, 9 and 10]. The differences can be motivated partly by the fact that the functions $f \in \boldsymbol{\mathcal{S}}[p]$ in the general case do not coincide with computable functions on any infinite computable domain. We sketch these differences in this final section.

The model of Fulk and Jain postulates — with several variations — that the learner outputs a sequence of guessed *programs* such that (a) there is an ascending sequence of sets A_n covering the whole domain $\boldsymbol{\mathcal{N}}$, (b) the n-th guess is correct on A_n and — in the case that density is involved in the definition — (c) the limit inferior of the densities of the sets A_n is greater or equal to given rational q.

This model already allows to approximate non-computable functions as, for example, the class $\boldsymbol{\mathcal{S}}$ of all functions f satisfying $f(x) = f(y)$, whenever the number of the trailing 0's of the decimal representations of x and y is the same. So, if $f \in \boldsymbol{\mathcal{S}}$, then $f(2) = f(3) = f(11111)$, $f(300) = f(900) = f(1100)$ and $f(7000) = f(9000)$ but $f(3)$ may be different from $f(20)$. So, if one knows $f(1)$, $f(10)$ and $f(100)$, then one knows already the value of f on 99.9% of the inputs; only the values of $f(x \cdot 1000)$ are unknown. The approximation which employs $f(x \cdot 10^k) = 0$, for the first k, where $f(10^k)$ has not yet been seen, coincides with f on a set of density $1 - 10^{-k}$. So the class $\boldsymbol{\mathcal{S}}$ is learnable by approximations with density 1.

The above example is also *predictable* with respect to all criteria defined in Section 1. This does not generalize since self-reference — as in [4] — might be employed. There are functions which, on input 0, output a program for the function itself. So one has that they are learnable with respect to all the criteria of Fulk and Jain: they are even “finitely” learnable. The problem is that the predictor does not know when it receives some faulty program from illegal input (which does not belong to any function in the learned class), but still has to predict some value then. So it is natural to ask whether some relation of the kind NV = PEx holds — where NV is the criterion to predict a function almost everywhere correctly by a total machine and PEx is to infer the function by a learner who on every input outputs only programs of total functions [4]. The next theorem shows that only one direction holds.

Theorem 22 *If some general class $\boldsymbol{\mathcal{S}}$ (possibly but not necessarily generated by some concept drift) is learnable with (uniform) density $q > 0$ in the manner defined by Fulk and Jain [7] and if this learner outputs for any input only programs for total functions, then $\boldsymbol{\mathcal{S}}$ is also predictable with (uniform) density q as defined in Definition 4 above.*

The converse does not hold.

Proof. Let φ_i denote the partial computable function computed by program i [14].

Suppose M learns with (uniform) density q in the manner defined by Fulk and Jain [7] and that M outputs, for any input, only programs for total functions. The corresponding prediction algorithm is the mapping N given by

$$N(f(0)f(1)\dots f(x)) = \varphi_{M(f(0)f(1)\dots f(x))}(x+1)$$

which is well-defined since M is total and outputs only total programs. Next it is shown that N learns every $f \in \boldsymbol{\mathcal{S}}$ with (uniform) density q. Let f be some function in $\boldsymbol{\mathcal{S}}$ and let $r < q$. Then there is a set A of (uniform) density r and a y such that, for all $x \geq y$, the program output by $M(f(0)f(1)\dots f(x))$ coincides with f on A. Thus, for $x \geq y$ and $x+1 \in A$, the program $M(f(0)...f(x))$ computes $f(x+1)$ correctly. Therefore the predictions of N are correct on every element of A greater than y. Since the (uniform) density of a set does not change if finitely many elements are removed, the set of all places where M predicts correctly has at least the (uniform) density r. Since this holds for all $r < q$ it follows that N predicts f with (uniform) density q. Since N does not depend on f it follows, that N predicts all $f \in \boldsymbol{\mathcal{S}}$ with (uniform) density q.

Next we consider the converse direction. Consider the class $\mathcal{C} =$

$\{f : (\forall x)\,(\forall y < 2x{+}1)\,[f(x^2{+}y) = f(x^2)]\}$. Now, a learner which predicts the value $f(z)$ for $f(z+1)$ is correct with (uniform) density 1 on each function of the class. We now show that, for every computable learner M, there is a function $f \in \mathcal{C}$ such that f does not coincide with any index output by M on an infinite set — this directly implies the non-learnability under all the criteria of Fulk and Jain [7].

Let $e_0, e_1, \ldots$ be an enumeration of all programs output by M on some input. All these programs compute total functions. One defines

$$f(x^2 + y) = 1 + \sum\nolimits_{n \leq x, z < (x+1)^2} \varphi_{e_n}(z)$$

where $0 \leq y < (x+1)^2 - x^2$ holds. The function f is in the class to be learned and satisfies, for $x \geq n$, $f(x^2 + z) > \varphi_{e_n}(x^2 + z)$. Thus it coincides with a given function φ_{e_n} on at most n^2 inputs. Thus M does not output any function which agrees with f on an infinite set. Therefore M does not learn f under the learning criteria given by Fulk and Jain [7]. ∎

A similar result to the previous one is the following which of course also holds with uniform density in place of density.

Theorem 23 *Assume that for a class $\boldsymbol{\mathcal{S}}$ (possibly but not necessarily generated by some concept drift) there is a recursively enumerable family of total computable functions $g_1, g_2, \ldots$ such that, for every $f \in \boldsymbol{\mathcal{S}}$, there is a g which coincides with f on a set of density greater than q (where q is rational). Then $\boldsymbol{\mathcal{S}}$ is learnable with density q.*

Proof. Let $d(i,z) = \frac{1}{z+1} \cdot |\{y \leq z : f(y) = g_i(y)\}|$ measure the density of the arguments y with $f(y) = g_i(y)$ below z. On input $f(0)f(1)\ldots f(x)$, the learner finds the least pair (i,j) such that $d(i,z) > q$ for all $z \in \{j, j+1, \ldots, x\}$ and then predicts $g_i(x)$. Since some g_i agrees with f on a set of density greater than q, there is a j such that $d(i,z) > q$, for all $z \geq j$. Thus, for all but finitely many x, the above algorithm uses a fixed pair (i,j) such that g_i and f agree with density at least q. Thus $\boldsymbol{\mathcal{S}}$ can be learned with density q. ▮

6 Some Concluding Remarks

Finally, we would like to point out a connection between our model and the mistake-bound learning model of Littlestone [12]. (We are grateful to an anonymous referee of COLT for enquiring about a connection.) Consider the setting from [12] in which a machine M predicts the values of a function f on a sequence of arguments $x_0, x_1, x_2, \ldots$ as follows. M is given x_0, M predicts the value of f at x_0, M is given $f(x_0)$, M is given x_1, M predicts the value of f at x_1, M is given $f(x_1)$, M is given x_2, M predicts the value of f at x_2, and so on. In general, *M learns a class $\boldsymbol{\mathcal{S}}$ of functions with mistake-bound c* iff M predicts, for each sequence $x_0, x_1, x_2, \ldots$ and each function $f \in \boldsymbol{\mathcal{S}}$, the function $i \rightarrow f(x_i)$ at all but at most c places correctly.

The sequence $x_0, x_1, x_2, \ldots$ can be arbitrary making this model difficult. Many of our basic definitions in the present paper depend quite essentially on order. We would, though, get our same results *mutatis mutandis* if we replaced the standard ordering of natural numbers by any *fixed* computable ordering.

For our remaining remarks we shall require the sequence $x_0, x_1, \ldots$ be *increasing.*

An example of a class of functions *then* so learnable with mistake-bound c is the class $\boldsymbol{\mathcal{S}}_c = \{$ decreasing $f : f(0) \leq c\}$.

The following interesting result can be shown. If a class $\boldsymbol{\mathcal{S}}$ is learnable with a mistake-bound of c and if $b \leq p$ for some constant permanence p, then $\boldsymbol{\mathcal{S}}[p]$ is learnable with frequency a out of b where $a = b - 2c - 1$. Furthermore, the class $\boldsymbol{\mathcal{S}}_c[p]$ cannot be learned with frequency $a + 1$ out of b; hence, the bound is tight.

Acknowledgement

We are especially grateful to Matthias Ott for his suggestions and encouragement regarding the present paper. We thank referees for several helpful comments. Arun Sharma was supported in part by Australian Research Council Grants A49600456 and A49803051. Frank Stephan was supported by the Deutsche Forschungsgemeinschaft (DFG) grant Am 90/6-2.

References

[1] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[2] P. Bartlett, S. Ben-David, and S. Kulkarni. Learning changing concepts by exploiting the structure of change. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory, Desenzano del Garda, Italy*, pages 131–139. ACM Press, July 1996.

[3] A. Blum and P. Chalasani. Learning switching concepts. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, Pennsylvania*, pages 231–242. ACM Press, 1992.

[4] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[5] M. Devaney and A. Ram. Dynamically adjusting concepts to accommodate changing contexts. In *Proceedings of AAAI*, page 1441, 1995.

[6] Y. Freund and Y. Mansour. Learning under persistent drift. In S. Ben-David, editor, *Proceedings of the Third European Conference on Computational Learning Theory* (EuroCOLT'97), volume 1208 of *Lecture Notes in Computer Science*, pages 94–108. Springer-Verlag, Berlin, 1997.

[7] M. Fulk and S. Jain. Approximate inference and scientific method. *Information and Computation*, 114(2):179–191, 1994.

[8] D. Helmbold and P. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14:27–46, 1994.

[9] S. Kaufmann and F. Stephan. Robust learning with infinite additional information. In S. Ben-David, editor, *Proceedings of the Third European Conference on Computational Learning Theory* (EuroCOLT'97), volume 1208 of *Lecture Notes in Computer Science*, pages 316–330. Springer-Verlag, Berlin, 1997.

[10] E. Kinber and T. Zeugmann. One-sided error probabilistic inductive inference and reliable frequency identification. *Information and Computation*, 92:253–284, 1991.

[11] M. Kubat. A machine learning based approach to load balancing in computer networks. *Cybernetics and Systems*, 23:389–400, 1992.

[12] N. Littlestone. *Mistake Bounds and Logarithmic Linear-Threshold Learning Algorithms.* PhD thesis, University of California, Santa Cruz, 1989.

[13] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer Verlag, Heidelberg, second edition, 1997.

[14] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967. Reprinted, MIT Press, 1987.

[15] J. Royer. Inductive inference of approximations. *Information and Control*, 70:156–178, 1986.

[16] C. Schnorr. *Zufälligkeit und Wahrscheinlichkeit.* Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1971.

[17] T. Shinohara, and S. Arikawa. Pattern Inference. In K. P. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, GOSLER Final Report, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 259–291. Springer-Verlag, Berlin, 1995.

[18] D. Widder. *Advanced Calculus.* Prentice-Hall, NJ, second edition, 1961.

[19] S. Wrobel. *Concept Formation and Knowledge Revision.* Kluwer Academic Publishers, 1994.