On Local Regret

Michael Bowling BOWLING@CS.UALBERTA.CA
Computing Science Department, University of Alberta, Edmonton, Alberta T6G2E8 Canada

Martin Zinkevich MAZ@YAHOO-INC.COM
Yahoo! Research, Santa Clara, CA 95051 USA

Abstract

Online learning aims to perform nearly as well as the best hypothesis in hindsight. For some hypothesis classes, though, even finding the best hypothesis offline is challenging. In such offline cases, local search techniques are often employed and only local optimality guaranteed. For online decision-making with such hypothesis classes, we introduce local regret, a generalization of regret that aims to perform nearly as well as only nearby hypotheses. We then present a general algorithm to minimize local regret with arbitrary locality graphs. We also show how the graph structure can be exploited to drastically speed learning. These algorithms are then demonstrated on a diverse set of online problems: online disjunct learning, online Max-SAT, and online decision tree learning.

1. Introduction

An online learning task involves repeatedly taking actions and, after an action is chosen, observing the result of that action. This is in contrast to offline learning where the decisions are made based on a fixed batch of training data. As a consequence offline learning typically requires i.i.d. assumptions about how the results of actions are generated (on the training data, and all future data). In online learning, no such assumptions are required. Instead, the metric of performance used is regret: the amount of additional utility that could have been gained if some alternative sequence of actions had been chosen. The set of alternative sequences that are considered defines the notion of regret. Regret is more than just a measure of performance, though, it also guides algorithms. For specific notions of regret, no-regret algorithms exist, for which the total regret is growing at worst sublinearly with time, hence their average regret goes to zero. These guarantees can be made with no i.i.d., or equivalent assumption, on the results of the actions.

One traditional drawback of regret concepts is that the number of alternatives considered must be finite. This is typically achieved by assuming the number of available actions is finite, and for practical purposes, small. In offline learning this is not at all the case: offline hypothesis classes are usually very large, if not infinite. There have been attempts to achieve regret guarantees for infinite action spaces, but these have all required assumptions to be made on the action outcomes (e.g., convexity or smoothness). In this work, we propose new notions of regret, specifically for very large or infinite action sets, while avoiding any significant assumptions on the sequence of action outcomes. Instead, the action set is assumed to come equipped with a notion of locality, and regret is redefined to respect this notion of locality. This approach allows the online paradigm with its style of regret guarantees to be applied to previously intractable tasks and hypothesis classes.

Appearing in *Proceedings of the 29^{th} International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

2. Background

For $t \in \{1, 2, \ldots\}$, let $a^t \in A$ be the action at time t, and $u^t : A \to \mathbb{R}$ be the utility function over actions at time t.

Requirement 1 *For all t,* $\max_{a,b \in A} |u^t(a) - u^t(b)| \leq \Delta$.

The basic building block of regret is the additional utility that could have been gained if some action b was chosen in place of action a: $R^T_{a,b} = \sum_{t=1}^{T} 1(a^t = a)\,(u^t(b) - u^t(a))$, where $1(\textit{condition})$ is equal to 1 when *condition* is true and 0 otherwise. We can use this building block to define the traditional notions of regret.

$$R^T_{\text{internal}} = \max_{a,b \in A} R^{T,+}_{a,b} \qquad R^T_{\text{swap}} = \sum_{a \in A} \max_{b \in A} R^{T,+}_{a,b} \quad (1)$$

$$R^T_{\text{external}} = \max_{b \in A} \left(\textstyle\sum_{a \in A} R^T_{a,b} \right)^+ \quad (2)$$

where $x^+ = \max(x, 0)$ so that $R^{T,+}_{a,b} = \max(R^T_{a,b}, 0)$. Internal regret (Hart & Mas-Colell, 2002) is the maximum utility that could be gained if one action had been chosen in place of some other action. Swap regret (Greenwald & Jafari, 2003) is the maximum utility gained if each action could be replaced by another. External regret (Hannan, 1957), which is the original pioneering concept of regret, is the maximum utility gained by replacing all actions with one particular action. This is the most relaxed of the three concepts, and while the others must concern themselves with $|A|^2$ possible regret values (for all pairs of actions) external regret only need worry about $|A|$ regret values. So although the guarantee is weaker, it is a simpler concept to learn which can make it considerably more attractive. These three regret notions have the following relationships.

$$R^T_{\text{internal}} \leq R^T_{\text{swap}} \leq |A| R^T_{\text{internal}} \qquad R^T_{\text{external}} \leq R^T_{\text{swap}} \tag{3}$$

Infinite Action Spaces. This paper considers situations where A is infinite. To keep the notation simple, we will use max operations over actions to mean suprema operations and summations over actions to mean the suprema of the sum over all finite subsets of actions. Since we will be focused on regret over a finite time period, there will only ever be a finite set of actually selected actions and, hence only a finite number of non-zero regrets, $R^T_{a,b}$. The summations over actions will always be thought to be restricted to this finite set.

None of the three traditional regret concepts are well-suited to A being infinite. Not only does $|A|$ appear in the regret bounds, but one can demonstrate that it is impossible to have no regret in some infinite cases. Consider $A = \mathbb{N}$ and let u^t be a step function, so $u^t(a) = 1$ if $a > y^t$ for some y^t and 0 otherwise. Imagine y^t is selected so that $\Pr[a^t > y^t | u^{1,\dots,T-1}, a^{1,\dots,T-1}] \leq 0.001$, which is always possible. Essentially, high utility is always just beyond the largest action selected. Now, consider $y^* = 1 + \max_{t \leq T} y^t$. In expectation $\frac{1}{T}\sum_{t=1}^T u^t(a_t) \leq 0.001$ while $\frac{1}{T}\sum_{t=1}^T u^t(y^*) = 1$ (i.e., there is large internal and external regret for not having played y^*,) so the average regret cannot approach zero.

Most attempts to handle infinite action spaces have proceeded by making assumptions on both A and u. For example, if A is a compact, convex subset of $\mathbb{R}^n$ and the utilities are convex with bounded gradient on A, then you can minimize regret even though A is infinite (Zinkevich, 2003). We take an alternative approach where we make use of a notion of locality on the set A, and modify regret concepts to respect this locality. Different notions of locality then result in different notions of regret. Although this typically results in a weaker form of regret for finite sets, it breaks all dependence of regret on the size of A and allows it to even be applied when A is infinite and u is an arbitrary (although still bounded) function. Wide range regret methods (Lehrer, 2003) can also bound regret with respect to a set of (countably) infinite "alternatives", but unlike our results, their asymptotic bound does not apply uniformly across the set, and uniform finite-time bounds depend upon a finite action space (Blum & Mansour, 2007).

3. Local Regret Concepts

Let $G = (V, E)$ be a directed graph on the set of actions, i.e., $V = A$. We do not assume A is finite, but we do assume G has bounded out-degree $D = \max_{a \in V} |\{b : (a, b) \in E\}|$. This graph can be viewed as defining a notion of locality. The semantics of an edge from a to b is that one should consider possibly taking action b in place of action a. Or rather, if there is no edge from a to b then one need not have any regret for not having taken action b when a was taken. By limiting regret only to the edges in this graph, we get the notion of local regret. Just as with traditional regret, which we will now refer to as global regret, we can define different variants of regret.

$$R^T_{\text{localinternal}} = \max_{(a,b) \in E} R^{T,+}_{a,b} \tag{4}$$

$$R^T_{\text{localswap}} = \sum_{a \in A} \max_{b:(a,b) \in E} R^{T,+}_{a,b} \tag{5}$$

Local internal and local swap regret just involve limiting regret to edges in G. Local external regret is more subtle and requires a notion of edge lengths. For all edges $(i, j) \in E$, let $c(i, j) > 0$ be the edge's positive length. Define $\mathrm{d}(a, b)$ to be the sum of the edge lengths on a shortest path from vertex a to vertex b, and $E^b = \{(i, j) \in E : d(i, j) = c(i, j) + d(j, b)\}$ to be the set of edges that are on any shortest path to vertex b.

$$R^T_{\text{localexternal}} = \max_{b \in A} \left(\sum_{(i,j) \in E^b} R^T_{i,j}/D \right)^+ \tag{6}$$

Global external regret considers changing all actions to some target action, regardless of locality or distance between the actions. In local external regret, only adjacent actions are considered, and so actions are only replaced with actions that take one step toward the target action. The factor of $1/D$ scales the regret of any one action by the out-degree, which is the maximum number of actions that could be one-step along a shortest path. This keeps local external regret on the same scale as local swap regret.

It is easy to see that these concepts hold the same relationships between each other as their global counterparts.

$$R^T_{\text{localinternal}} \leq R^T_{\text{localswap}} \leq |A| R^T_{\text{localinternal}} \tag{7}$$

$$R^T_{\text{localexternal}} \leq R^T_{\text{localswap}} \tag{8}$$

More interestingly, in complete graphs where there is an edge between every pair of actions (all with unit lengths) and so everything is local, we can exactly equate global and local regret.

Theorem 1 *If G is a complete graph with unit edge lengths then, $R^T_{\text{localinternal}} = R^T_{\text{internal}}$; $R^T_{\text{localswap}} = R^T_{\text{swap}}$; and $R^T_{\text{localexternal}} = R^T_{\text{external}}/D$.*

The proofs of the paper's theorems are not included for space reasons. When there is a useful insight, we discuss the proof techniques and implications. The full proofs can be found in the longer version of this work available as a technical report (Bowling & Zinkevich, 2012).

So our concepts of local regret match up with global regret when the graph is complete. Of course, we are not really interested in complete graphs, but rather more intricate locality structures with a large or infinite number of vertices, but a small out-degree. Before going on to present algorithms for minimizing local regret, we consider possible graphs for three different online decision tasks to illustrate where the graphs come from and what form they might take.

Example 1 (Online Max-3SAT) Consider an online version of Max-3SAT. The task is to choose an assignment for n boolean variables: $A = \{0,1\}^n$. After an assignment is chosen a clause is observed; the utility is 1 if the clause is satisfied by the chosen assignment, 0 otherwise. Note that $|A| = 2^n$ which is computationally intractable for global regret concepts if n is even moderately large. One possible locality graph for this hypothesis class is the hypercube with an edge from a to b if and only if a and b differ on the assignment of exactly one variable, and all edges have unit lengths. So the out-degree D for this graph is only n. Local regret, then, corresponds to the regret for not having changed the assignment of just one variable. In essence, minimizing this concept of regret is the online equivalent of local search (e.g., WalkSAT (Selman et al., 1993)) on the maximum satisfiability problem, an offline task where all of the clauses are known up front.

Example 2 (Online Disjunct Learning) Consider a boolean online classification task where input features are boolean vectors $x \in \{0,1\}^n$ and the target y is also boolean. Consider $A = \{0,1\}^n$, to be the set of all disjuncts such that $a \in A$ corresponds to the disjunct $x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}$ where $i_{1 \leq j \leq k}$ are all of the k indices of a such that $a_{i_j} = 1$. In this online task, one must repeatedly choose a disjunct and then observe an instance which includes a feature vector and the correct response. There is a utility of 1 if the chosen disjunct over the feature vector results in the correct response; 0 otherwise. Although a very different task, the action space $A = \{0,1\}^n$ is the same as with Online Max-SAT and we can consider the same locality structure as that proposed for disjuncts: a hypercube with unit length edges for adding or removing a single variable to the disjunction. And as before $|A| = 2^n$ while $D = n$.

Example 3 (Online Decision Tree Learning) Imagine the same boolean online classification task for learning disjuncts, but the hypothesis class is the set of all possible decision trees. The number of possible decision trees for n boolean variables is more than a staggering 2^{2^n}, which for any practical purpose is infinite. We can construct a graph structure that mimics the way decision trees are typically constructed offline, such as with C4.5 (Quinlan, 1993). In the graph G, add an edge from one decision tree to another if and only if the latter can be constructed by choosing any node (internal or leaf) of the former and replacing the subtree rooted at the node with a decision stump or a label. There is one exception: you cannot replace a non-leaf subtree with a stump splitting on the same variable as that of the root of the subtree. Edges that replace a subtree with a label have length 1, while edges replacing a subtree with a stump (being a more complex change) have distance 1.1. So, we have local regret for not having further refined a leaf or collapsing a subtree to a simpler stump or leaf. Notice that the graph edges in this case are not all symmetric (viz., collapsing edges). In essence, this is the online equivalent of tree splitting algorithms. While $|A| \geq 2^{2^n}$, the out-degree is no more than $(n+1)2^{n+1}$. The maximum size of the out-degree still appears disconcertingly large, and we will return to this issue in Section 5 where we show how we can exploit the graph structure to further simplify learning.

4. An Algorithm for Local Swap Regret

We now present an algorithm for minimizing local swap regret, similar to global swap regret algorithms (Hart & Mas-Colell, 2002; Greenwald & Jafari, 2003), but with substantial differences. The algorithm essentially chooses actions according to the stationary distribution of a Markov process on the graph, with the transition probabilities on the edges being proportional to the accumulated regrets. However there are two caveats that are needed for it to handle infinite graphs: it is prevented from playing beyond a particular distance from a designated root vertex, and there is an internal bias towards the actual actions chosen.

Formally, let root be some designated vertex. Define d_1 to be the unweighted shortest path distance between two vertices. Define the level of a vertex as its distance from root: $\mathcal{L}(v) = \mathrm{d}_1(\text{root}, v)$. Note that, $\mathcal{L}(\text{root}) = 0$, and $\forall (i,j) \in E, \mathcal{L}(j) \leq \mathcal{L}(i) + 1$. All of the algorithms in this paper take a parameter L, and will never choose actions at a level greater than L. In addition, the algorithms all maintain values $\tilde{R}^t_{i,j}$ (which are biased versions of $R^t_{i,j}$) and use these to compute π^t_j, the probability of choosing action j at time t. These probabilities are always computed according to the following requirement, which is a generalization of (Hart & Mas-Colell, 2002; Greenwald & Jafari, 2003).

Requirement 2 *Given a parameter L, for all $t \leq T$, and some $\tilde{R}^{t,+}_{i,j}$ let π^{t+1} be such that*

(a) $\sum_{j \in V} \pi^{t+1}_j = 1$, *and* $\forall j \in V$, $\pi^{t+1}_j \geq 0$

(b) $\forall j \in V$ *such that* $\mathcal{L}(j) > L$, $\pi^{t+1}_j = 0$.

(c) $\forall j \in V$ *such that* $1 \leq \mathcal{L}(j) \leq L$, $\pi^{t+1}_j = \sum_{i:(i,j)\in E} (\tilde{R}^{t,+}_{i,j}/M)\pi^{t+1}_i + (1 - \sum_{k:(j,k)\in E} \tilde{R}^{t,+}_{j,k}/M)\pi^{t+1}_j$

(d) $\pi^{t+1}_{\text{root}} = \sum_{i:(i,\text{root})\in E} (\tilde{R}^{t,+}_{i,\text{root}}/M)\pi^{t+1}_i + \sum_{j:\mathcal{L}(j)=L+1} \sum_{i:(i,j)\in E} (\tilde{R}^{t,+}_{i,j}/M)\pi^{t+1}_i + (1 - \sum_{j:(\text{root},j)\in E} \tilde{R}^{t,+}_{\text{root},j}/M)\pi^{t+1}_{\text{root}}$

(e) If there exists $j \in V$ such that $\pi^{t+1}_j > 0$ and $\sum_{k:(j,k)\in E} \tilde{R}^{t,+}_{j,k} = 0$, then for all $j \in V$ where $\pi^{t+1}_j > 0$, $\sum_{k:(j,k)\in E} \tilde{R}^{t,+}_{j,k} = 0$, and we call such a π^{t+1} degenerate.

where $M = \max_{(i,j)\in E} \tilde{R}^{t,+}_{i,j}$. These conditions require π^{t+1} to be the stationary distribution of the transition function whose probabilities on outgoing edges are proportional to their biased positive regret, with the root vertex as the starting state, and all outgoing transitions from vertices in level L going to the root vertex instead.

Definition 2 *(b, L)-regret matching is the algorithm that initializes $\tilde{R}^0_{i,j} = 0$, chooses actions at time t according to a distribution π^t that satisfies Requirement 2 and after choosing action i and observing u^t updates $\tilde{R}^t_{i,j} = \tilde{R}^{t-1}_{i,j} + (u^t(j) - u^t(i) - b)$ for all j where $(i, j) \in E$, and for all other $(k, l) \in E$ where $k \neq i$, $\tilde{R}^t_{k,l} = \tilde{R}^{t-1}_{k,l}$.*

There are two distinguishing factors of our algorithm from (Hart & Mas-Colell, 2002; Greenwald & Jafari, 2003): $\tilde{R} \neq R$, and past a certain distance from the root, we loop back. $\tilde{R}$ differs from R by the bias term, b. This term can be thought of as a bias toward the action selected by the algorithm. This is *not* the same as approaching the negative orthant with a margin for error. This small amount is only applied to the action taken, which is very different from adding a small margin of error to *every* edge.

Theorem 3 *For any directed graph with maximum out-degree D and any designated vertex* root*, $(\Delta/(L+1), L)$-regret matching, after T steps, will have expected local swap regret no worse than,*

$$\frac{1}{T} E[R^T_{\text{localswap}}] \leq \frac{\Delta}{L+1} + \frac{\Delta\sqrt{D|E_L|}}{\sqrt{T}} \qquad (9)$$

where $E_L = \{(i, j) \in E | \mathcal{L}(i) \leq L\}$.

The overall structure of the proof is similar to (Blackwell, 1956; Hart & Mas-Colell, 2002; Greenwald & Jafari, 2003) with a few significant changes. As with most algorithms based on Blackwell, if there is an action you do not regret taking, playing that action the next round is "safe". If not, the key quantity in the proof is a flow $f_{i,j} = \pi^{t+1}_i \tilde{R}^{t,+}_{i,j}$ for each edge. On most of the graph, the incoming flow is equal to the outgoing flow for each node in levels 1 to L. Since all the flow out from the nodes on one level is equal to the flow into the next, the total flow into (and out of) each level is equal. Thus, the flow out of the last level is only $1/(L+1)$ of the total flow on all edges since there are $L+1$ levels, including the root.

Traditionally, we wish to show that the incoming flow of an action times the utility minus the outgoing flow of an action times the utility summed over all nodes is nonpositive, and then Blackwell's condition holds. In traditional proofs, for any given node, the flow in and out are equal, so regardless of the utility, they cancel. For our problem, the flow out of the last level is really a flow into the $(L+1)$st level, not the zeroeth level, so the difference in utilities between the zeroeth level and the $(L+1)$st level creates a problem. On the other hand, because we subtract b from whatever action we select, we get to subtract b times the total flow. Since exactly $1/(L+1)$ fraction of the flow is going into the $(L+1)$st level, these two discrepancies from the traditional approach exactly cancel. The second term of Equation (9) is a result of the traditional Blackwell approach. In the final analysis, we must account for the amount b we subtract from the regret each round. This means that if we get $\tilde{R}$ to approach the negative orthant, we only have bT local swap regret left. This is the first term of Equation (9).

5. Exploiting Locality Structure

The local swap regret algorithm in the previous section successfully drops all dependence on the size of the action set and thus can be applied even for infinite action sets. However, the appearance of $|E_L|$ in the bound in Theorem 3 is undesirable as $|E_L| \in O(D^L)$, and L is more likely to be 100 than 2, in order to keep the first term of the bound low. The bound, therefore, practically provides little beyond an asymptotic guarantee for even the simplest setting of Example 1. In this section, we will appeal to (i) the structure in the locality graph, and (ii) local external regret to achieve a more practical regret bound and algorithm.

Cartesian Product Graphs. We begin by considering the case of G having a very strong structure, where it can be entirely decomposed into a set of product graphs. In this case, we can show that by independently minimizing local regret in the product graphs we can minimize local regret in the full graph.

Theorem 4 *Let G be a Cartesian product of graphs, $G = G_1 \otimes \ldots \otimes G_k$. Let $R^{T,l}_{\text{localexternal}}$ be the measured external*

regret on the l^{th} component graph, where the action at time t is the l^{th} component of a^t and regret is on the edges in G_l that transform the l^{th} component. Then, $R^T_{\text{localexternal}} \leq \sum_{l=1}^{k} R^{T,l}_{\text{localexternal}}$.

The implication is that we if we apply independent regret minimization to each factor of our product graph, we can minimize local external regret on the full graph. For example, consider the hypercube graphs from Example 1 and 2. By applying n independent external regret algorithms (the component graphs in this case are 2-vertex complete graphs), the overall local external regret for the graph is at most n times bigger than the factors' regrets, so under regret matching it is bounded by $n\Delta\sqrt{2}/\sqrt{T}$. Hence, we are able to handle an exponentially large graph (in n) with local external regret only growing linearly (in n). If the component graphs are not complete graphs, then we can simply apply our local swap regret algorithm from the previous section to the graph factors, which minimizes local external regret as well.

Color Regret. Cartesian product graphs are a powerful, but not very general structure. We now substantially generalize the product graph structure, which will allow us to achieve a similar simplification for very general graphs, such as the graph on decision trees in Example 3. The key insight of product graphs is that for any vertex b, an edge moves toward b if and only if its corresponding edge in its component graph moves toward b_l. In other words, either all of the edges that correspond to some component edge will be included in the external regret sum, or none of the eges will. We can group together these edges and only worry about the regret of the group and not its constituents. We generalize this fact to graphs which do not have a product structure.

Definition 5 *An edge-coloring $\mathbf{C} = \{C_i\}_{i=1,2,\ldots}$ for an arbitrary graph G with edge lengths is a partition of E: $C_i \subseteq E$, $\bigcup_i C_i = E$, and $C_i \bigcap C_j = \emptyset$. We say that $\mathbf{C}$ is admissble if and only if for all $b \in V$, $C \in \mathbf{C}$, and $(i,j),(i',j') \in C$, $\mathrm{d}(i,b) = c(i,j) + \mathrm{d}(i,b) \Leftrightarrow \mathrm{d}(i',b) = c(i',j') + \mathrm{d}(j',b)$. In other words, for any arbitrary target, all of the edges with the same color are on a shortest path, or none of the edges are.*

We now consider treating all of the edges of the same color as a single entity for regret. This gives us the notion of local colored regret.

$$R^T_{\text{localcolor}} = \sum_{C \in \mathbf{C}} \left(\sum_{(i,j) \in C} R^T_{i,j} \right)^+ \tag{10}$$

Theorem 6 *If $\mathbf{C}$ is admissible then $R^T_{\text{localexternal}} \leq R^T_{\text{localcolor}}/D$.*

So by minimizing local colored regret, we minimize local external regret. The natural extension of our local swap regret algorithm from the previous section results in an algorithm that can minimize local colored regret.

Definition 7 *$(b, L, \mathbf{C})$-colored-regret-matching is the algorithm that initializes $\tilde{R}^0_C = 0$, for all $C \in \mathbf{C}$, chooses actions at time t according to a distribution π^t that satisfies Requirement 2 with $\tilde{R}^t_{i,j} \equiv \tilde{R}^t_{c(i,j)}$, and after choosing action i and observing u^t at time t for all $C \in \mathbf{C}$ updates $\tilde{R}^t_C = \tilde{R}^{t-1}_C + \sum_{j:(i,j)\in C}(u^t(j) - u^t(i) - b)$.*

Theorem 8 *For an arbitrary graph G with maximum degree D, arbitrarily chosen vertex* root*, and edge coloring $\mathbf{C}$, $(\Delta/(L+1), L, \mathbf{C})$-colored-regret matching applied after T steps will have expected local colored regret no worse than,*

$$\frac{1}{T} E[R^T_{\text{localcolor}}] \leq \frac{\Delta D}{L+1} + \frac{\Delta\sqrt{D|C_L|}}{\sqrt{T}}$$

where $C_L = \{C \in \mathbf{C} | \exists (i,j) \in C \text{ s.t. } \mathcal{L}(i) \leq L\}$.

The consequence of this bound depends upon the number of colors needed for an admissible coloring. Very small admissible colorings are often possible. The hypercube graph needs only $2n$ colors to give an admissible coloring, which is exponentially smaller than the total number of edges, $n2^n$. We can also find a reasonably tight coloring for our decision tree graph example, despite being a complex asymmetric graph.

Example 4 (Colored Decision Tree Learning)
Reconsider Example 3. Recall that an edge exists between one decision tree and another if the latter can be constructed from the former by replacing a subtree at any node (internal or leaf) with a label (edge length 1) or a stump (edge length 1.1). We will color this edge with the pair: (i) the sequence of variable assignments that is required to reach the node being replaced, and (ii) the stump or label that replaces it. This coloring is admissible. We can see this fact by considering a color: the sequence of variable assignments and resulting stump or label. If this color is consistent with the target decision tree (i.e., the sequence exists in the target decision tree, and the variable of the added stump matches the variable split on at that point in the target decision tree) then the color must move you closer to the target tree.

6. Experimental Results

The previous section presented algorithms that minimize local swap and local external regret (by minimizing local colored regret). The regret bounds have no dependence on the size of the graph beyond the graph's degree, and so provide a guarantee even for infinite graphs. We now explore

these algorithms' practicality as well as illustrate the generality of the concepts by applying them to a diverse set of online problems. The first two tasks we examine, online Max-3SAT and online decision tree learning, have not previously been explored in the online setting. The final task, online disjunct learning, has been explored previously, and will help illustrate some drawbacks of local regret.

In all three domains we examine two algorithms. The first minimizes local swap regret by applying $(\Delta/(L+1), L)$-regret matching with L chosen specifically for the problem. This will be labeled "Local Swap". The second focuses on local external regret by using a tight, admissible edge-coloring and applying $(\Delta/(L+1), L, \mathbf{C})$-colored-regret matching. This will be labeled simply "Local External".

Online Max-3SAT. First, we consider Example 1. We randomly constructed problem instances with $n = 20$ boolean variables and 201 clauses each with 3 literals. On each timestep, the algorithms selected an assignment of the variables, a clause was chosen at random from the set, and the algorithm received a utility of 1 if the assignment satisfied the clause, 0 otherwise. This was repeated for 1000 timesteps. The locality graph used was the n-dimensional hypercube from Example 1. The admissible coloring used to minimize local external regret was the $2n$ coloring that has two colors per variable (one for turning the variable on, and one for turning the variable off). In both cases we set $L = \infty$ and $b = 0$, since the bounds do not depend on L once it exceeds 20. This also achieved the best performance for both algorithms. The average results over 200 randomly constructed sets of clauses are shown in Figure 1, with 95% confidence bars.

Figure 1 (a) shows the time-averaged colored regret of the two algorithms, to demonstrate how well the algorithms are actually minimizing regret. Both are decreasing over time, while external regret is decreasing much more rapidly. As expected, swap regret may be a stronger concept, but it is more difficult to minimize. The local external regret algorithm after only one time step can have regret for not having made a particular variable assignment, while local swap regret has to observe regret for this assignment from every possible assignment of the other variables to achieve the same result. This is further demonstrated by the number of regret values each algorithm is tracking: local external regret on average had 34 non-zero regret values, while local swap regret had 4200 non-zero regret values. In summary, external regret provides a powerful form of generalization. Figure 1 (b) shows the fraction of the previous 100 clauses that were satisfied. Two baselines are also presented. A random choice of variable assignments can satisfy $\frac{7}{8}$ of the clauses in expectation. We also ran WalkSAT (Selman et al., 1993) offline on the set of 201 clauses, and on average it was able to satisfy all but 4% of the clauses, which

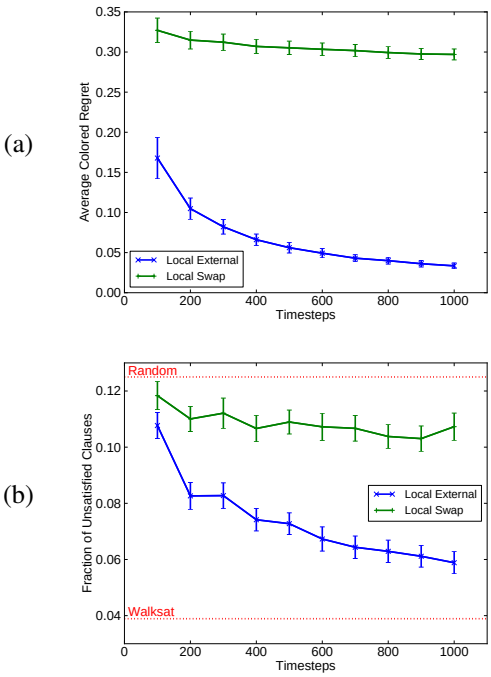

Figure 1. Results for Online Max-3SAT: (a) regret, (b) fraction of unsatisfied clauses.

gives an offline lower bound for what is possible. Both substantially outperformed random, with the external regret algorithm nearing the performance of the offline WalkSat.

Online Decision Tree Learning. Second, we consider Example 3. We took three datasets from the UCI Machine Learning Repository (each with categorical inputs and a large number of instances): nursery, mushroom, and king-rook versus king-pawn (Frank & Asuncion, 2010). The categorical attributes were transformed into boolean attributes (which simplified the implementation of the locality graphs) by having a separate boolean feature for each attribute value.[1] We made the problems online classification tasks by sampling five instances at random (with replacement) for each timestep, with the utility being the number classified correctly by the algorithm's chosen decision tree. This was repeated for 1000 timesteps, and so the algorithms classified 5,000 instances in total. The locality graph used was the one described in Example 3. The tight coloring used to minimize local external regret was the one described in Example 4. L was set to 3 for local swap regret, and 100 for local external regret, as this achieved the best performance. Even with the far larger graph, the external regret algorithm was observing nearly one-eighth of the number of non-zero regret values observed by the lo-

[1] As a result, there were $n = 28$ features for nursery, 118 features for mushroom, and 74 features for king-rook versus king-pawn.

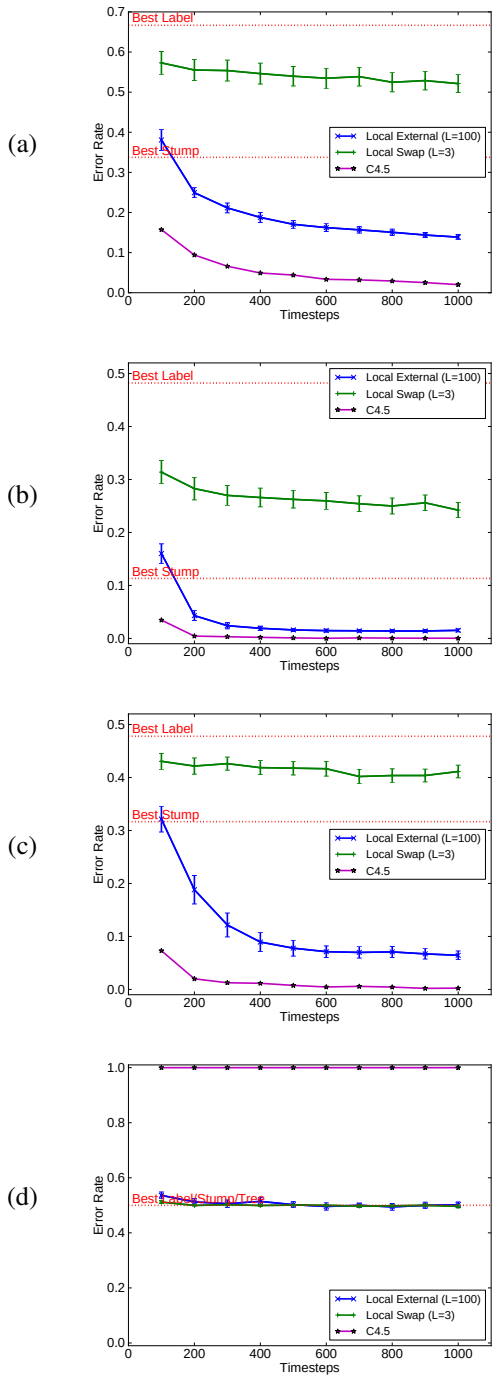

Figure 2. Results for online decision tree learning on three UCI datasets: (a) Nursery, (b) Mushroom, (c) King-Rook/King-Pawn; and (d) a simple sequence of alternating labels.

cal swap algorithm. The average results over 50 trials are shown in Figure 2(a)-(c) with 95% confidence bars.

The graphs show the average fraction of misclassified instances over the previous 100 timesteps. Two baselines are also plotted: the best single label (i.e., the size of the majority class) and the best decision stump. Both regret algorithms substantially improved on the best label, and local external regret was selecting trees substantially better than the best stump. As a further baseline, we ran the batch algorithm C4.5 in an online fashion, by retraining a decision tree after each timestep using all previously observed examples. C4.5's performance was impressive, learning highly accurate trees after observing only a small fraction of the data. However, C4.5 has no regret guarantees. As with any offline algorithm used in an online fashion, there is an implicit assumption that the past and future data instances are i.i.d.. In our experimental setup, the instances were i.i.d., and as a result C4.5 performed very well. To further illustrate this point, we constructed a simple online classification task where instances with identical attributes were provided with alternating labels. The best label (as well as the single best decision tree) has a 50% accuracy. C4.5 when trained on the previously observed instances, misclassifies every single instance. This is shown along with local regret algorithms in Figure 2 (d).

Online Disjunct Learning. Finally, we examine online disjunct learning as described in Example 2. This task has received considerable attention, notably the celebrated Winnow algorithm (Littlestone, 1988), which is guaranteed to make a finite number of mistakes if the instances can be perfectly classified by some disjunction. Furthermore, the number of mistakes Winnow2 makes, when no disjunction captures the instances, can be bounded by the number of attribute errors (i.e., the number of input attributes that must be flipped to make the disjunction satisfy the instance) made by the best disjunction. In these experiments we compare our algorithms' performance to that of Winnow2.

We looked at two learning tasks. In the first, we generated a random disjunction over $n = 20$ boolean variables, where a variable was independently included in the disjunction with probability $4/n$. Instances were created with uniform random assignments to all of the variables, with a label being true if and only if the chosen disjunct is true for the instance's assignment. In the second case, we chose instances uniformly at random from a constructed set of 21 instances: one for each variable with that variable (only) set to true and the label being true, and one with all of the variables assigned the value of true and the label being false. We call this task Winnow Killer. For both tasks, the n-dimensional hypercube from Example 1 was used as the locality graph with the $2n$ coloring as our admissible coloring, and $L = \infty$ and $b = 0$. The average results over 50 trials are shown in Figure 3, with 95% confience bars.

The graphs plot error rates over the previous 100 instances. Three baselines are plotted: randomly assigning a label (guaranteed to get half of the instances correct on expec-

(a)

(b)

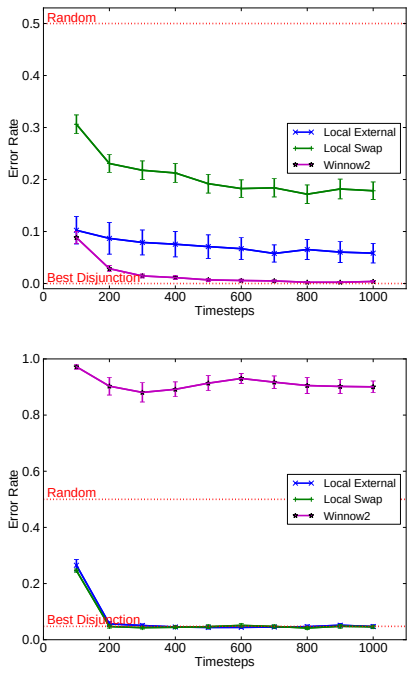

Figure 3. Results for online disjunct learning: (a) random disjunct, (b) Winnow Killer.

tation), the best disjunct (which makes no mistakes for random disjunctions and makes $\frac{1}{21}$ mistakes on the Winnow Killer task), and Winnow2. Figure 3 (a) shows the results on random disjunctions. Winnow2 is guaranteed to make a finite number of mistakes and indeed its error rate drops to zero quickly. The local regret concepts, though, have difficulties with random disjunctions. The reason can be easily seen for the case of local external regret. Suppose the first instance is labeled true; the algorithm now has regret for all of the variables that were true in that instance (some of these will be in the target disjunction, but many will not). These variables will now be included in the chosen disjunction for a very long time, as the only regret that one can have for not removing them is if their assignment was the sole reason for misclassifying a false instance. In other words, the problem is that there's no regret for not removing multiple variables simultaneously as this is not a local change. Winnow2, though, also has issues. It performs very poorly in the Winnow Killer task (in fact, if the instances were ordered it could be made to get every instance wrong), as shown in Figure 3 (b). Since the mistake bound for Winnow2 is with respect to the number of attribute errors, a single mistake by the best disjunction can result in n mistakes by Winnow2. A further issue with Winnow is that while its peformance is tied to the performance of disjunctions, its own hypothesis class is not disjunctions but a thresholded linear function, whereas local regret is playing in the same class of hypotheses that it comparing against.

7. Conclusion

We introduced a new family of regret concepts based on restricting regret to only nearby hypotheses using a locality graph. We then presented algorithms for minimizing these concepts, even when the number of hypotheses are infinite. Further we showed that we can exploit structure in the graph to achieve tighter bounds and better performance. These new regret concepts mimic local search methods, which are common approaches to offline optimization with intractably hard hypothesis spaces. As such, our concepts and algorithms allows us to make online guarantees, with a similar flavor to their offline counterparts, with these hypothesis spaces.

Acknowledgements

This work was supported by NSERC and Yahoo! Research, where the first author was a visiting scientist at the time the research was conducted.

References

Blackwell, D. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6:1–8, 1956.

Blum, A. and Mansour, Y. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, 2007.

Bowling, M. and Zinkevich, M. On local regret. Technical Report TR12-04, University of Alberta, 2012.

Frank, A. and Asuncion, A. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

Greenwald, A. and Jafari, A. A general class of no regret learning algorithms and game-theoretic equilibria. In *Proceedings of the Sixteenth Annual Conference on Learning Theory*, 2003.

Hannan, J. Approximation to bayes risk in repeated plays. In Dresher, M., Tucker, A., and Wolfe, P. (eds.), *Contributions to the Theory of Games*, volume 3, pp. 97–139. Princeton University Press, 1957.

Hart, S. and Mas-Colell, A. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):181–200, 2002.

Lehrer, E. A wide range no-regret theorem. *Games and Economic Behavior*, 42:101–115, 2003.

Littlestone, N. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufman Publishers, 1993.

Selman, B., Kautz, H., and Cohen, B. Local search strategies for satisfiability testing. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, October 1993.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Twentieth International Conference on Machine Learning*, pp. 928–936, 2003.