A. Appendix

A.1. Proof of Corollary 1

Restatement of Corollary 1:
Assume we have a family of functions \( F_\Theta \), a KWIK-learning algorithm \( \text{KWIK} \) for \( F_\Theta \), and a fixed-state optimization algorithm \( \text{FixedStateOpt} \). Then there exists a no-regret algorithm for the MAB problem on \( F_\Theta \).

Proof. Let \( A(\epsilon, \delta) \) denote Algorithm 1 when parameterized by \( \epsilon \) and \( \delta \). We construct a no-regret algorithm \( A^* \) for the MAB problem on \( F_\Theta \) that operates over a series of epochs. On the start of epoch \( i \), \( A^* \) simply runs a fresh instance of \( A(\epsilon_i, \delta_i) \), and does so for \( \tau_i \) rounds. We will describe how \( \epsilon_i, \delta_i, \tau_i \) are chosen.

First let \( e(T) \) denote the number of epochs that \( A^* \) starts after \( T \) rounds. Let \( \gamma_i \) be the average regret suffered on the \( i \)th epoch. In other words, if \( x^{i,t}(a^{i,t}) \) is the \( i \)th state (action) in the \( i \)th epoch, then

\[
\gamma_i = E \left( \frac{1}{T_i} \sum_{t=1}^{T_i} \max_{a^{i,t} \in A} f_\theta(x^{i,t}, a^{i,t}) - f_\theta(x^{i,t}, a^{t,i}) \right).
\]

We therefore can express the average regret of \( A^* \) as:

\[
R_{A^*}(T) = \frac{1}{T} \sum_{i=1}^{e(T)} \tau_i \gamma_i \tag{1}
\]

From Theorem 1, we know there exists a \( T_i \) and choices for \( \epsilon_i \) and \( \delta_i \) so that \( \gamma_i < 2^{-i} \) so long as \( T_i \geq T_i \). Let \( T_1 = T_1 \), and \( \tau_1 = 2\gamma_1 \). These choices for \( \tau_1, \epsilon_1 \) and \( \delta_1 \) guarantee that \( \tau_{i-1} \leq \tau_i / 2 \), and also \( \gamma_i < 2^{-i} \). Applying these facts respectively to Equation 1 allows us to conclude that:

\[
R_{A^*}(T) \leq \frac{1}{T} \sum_{i=1}^{e(T)} 2^{-(e(T)-i)} \tau_{e(T)} \gamma_i < \frac{1}{T} \sum_{i=1}^{e(T)} 2^{-e(T)} \tau_{e(T)} \leq e(T)2^{-e(T)}
\]

Theorem 1 also implies that \( e(T) \to \infty \) as \( T \to \infty \), and so \( A^* \) is indeed a no regret algorithm.

A.2. Proof of Corollary 2

Restatement of Corollary 2:
If the don’t-know bound of KWIK is \( B(\epsilon, \delta) = O(\epsilon^{-d}\log^{d-1}) \) for some \( d > 0, k \geq 0 \) then there are choices of \( \epsilon, \delta \) so that the average regret of Algorithm 1 is:

\[
O\left( \left( \frac{1}{T} \right)^{\frac{1}{2} + \frac{1}{2\delta} + \frac{1}{2d}} \log k \right)
\]

Proof. Taking \( \epsilon = \left( \frac{1}{T} \right)^{\frac{1}{2} + \frac{1}{2\delta} + \frac{1}{2d}} \) and \( \delta = \frac{1}{T} \) in Equation 2 in the proof of Theorem 1 suffices to prove the corollary.

A.3. Proof of Theorem 2

We proceed to give a the proof of Theorem 2 in complete rigor. We will first give a more precise construction of the class of models \( F_\theta \) satisfying the conditions of the theorem.

Restatement of Theorem 2:
There exists a class of models \( F_\theta \) such that

- \( F_\theta \) is fixed-state optimizable;
- There is an efficient algorithm \( A \) such that on an arbitrary sequence of \( T \) trials \( z^t \), \( A \) makes a prediction \( y^t \) of \( y^t = f_\theta(z^t) \) and then receives \( y^t \) as feedback, and the total regret \( \sum_{t=1}^{T} |y^t - y^t| \) is sub-linear in \( T \) (thus we have only no-regret supervised learning instead of the stronger KWIK);
- Under standard cryptographic assumptions, there is no polynomial-time algorithm for the no-regret MAB problem for \( F_\Theta \).

Let \( Z_n = \{0, ..., n-1\} \). Suppose that \( \Theta \) parameterizes a family of cryptographic trapdoor functions \( H_\Theta \) (which we will use to construct \( F_\Theta \)). Specifically, each \( \theta \) consists of a “public” and “private” part so that

\[
\theta = (\theta_{\text{pub}}, \theta_{\text{pri}}), \quad H_\Theta = \{ h_\theta : Z_n \to Z_n \}. \]

The cryptographic guarantee ensured by \( H_\Theta \) is summarized in the following definition.

Definition 1. Let \( d = \lfloor \log |Z_n| \rfloor \). Any family of cryptographic trapdoor functions \( H_\Theta \) must satisfy the following conditions:

- (Efficiently Computable) For any \( \theta \), knowing just \( \theta_{\text{pub}} \) gives an efficient (polynomial in \( d \)) algorithm for computing \( h_\theta(a) \) for any \( a \in Z_n \).
- (Not Invertible) Let \( k \) be chosen uniformly at random from \( Z_n \). Let \( A \) be an efficient (randomized) algorithm that takes \( \theta_{\text{pub}} \) and \( h_\theta(k) \) as input (but not \( \theta_{\text{pri}} \)), and outputs an \( a \in Z_n \). There is no polynomial \( q \) such that \( P(h_\theta(k) = h_\theta(a)) \geq 1/q(d) \).
Depending on the family of trapdoor functions, the second condition usually holds under an assumption that some problem is intractable (e.g. prime factorization).

We are now ready to describe \((\mathcal{F}_\theta, \mathcal{A}, \mathcal{X})\). Fix \(n\), and let \(\mathcal{X} = \mathbb{Z}_n\) and \(\mathcal{A} = \mathbb{Z}_n \cup \{a^*\}\). For any \(h_\theta \in H_\theta\), let \(h_\theta^{-1}\) denote the inverse function to \(h_\theta\). Since \(h_\theta\) may be many-to-one, for any \(y\) in the image of \(h_\theta\), arbitrarily define \(h_\theta^{-1}(y)\) to be any \(x\) such that \(h_\theta(x) = y\).

We will define the behavior of each \(f_\theta \in \mathcal{F}_\theta\) in what follows. First we will define a family of functions \(G_\theta\). The behavior of each \(g_\theta\) will be essentially identical to that of \(f_\theta\), and for the purposes of understanding the construction, it is useful to think of them as being exactly identical.

The behavior of \(g_\theta\) on states \(x \in \mathbb{Z}_n\) is defined as follows. Given \(x\), to get the latest payoff of 1, an algorithm must invert \(h_\theta\). In other words, \(g_\theta(x, a) = 1\) only if \(h_\theta(a) = x\) (for \(a \in \mathbb{Z}_n\), and not equal to the “special” action \(a^*\)). For any other \(a \in \mathbb{Z}_n\), \(g_\theta(x, a) = 0\).

On action \(a^*\), \(g_\theta(x, a^*)\) reveals the location of \(h_\theta^{-1}(x)\). Specifically \(g_\theta(x, a^*) = \lfloor \frac{0.5}{1 + h_\theta^{-1}(x)} \rfloor\) if \(x\) has an inverse and \(g_\theta(x, a^*) = 0\) if \(x\) is not in the image of \(h_\theta\). It’s useful to pause here, and consider the purpose of the construction. Assume that \(\theta_{\text{pub}}\) is known. Then if \(x\) and \(a\) (\(a \in \mathbb{Z}_n\) are presented simultaneously in the supervised learning setting, it’s easy to simply check if \(h_\theta(x) = a\), making accurate predictions. In the fixed-state optimization setting, querying \(a^*\) presents the algorithm with all the information it needs to find a maximizing action. However, in the bandit setting, if a new \(x\) is being drawn uniformly at random and presented to the algorithm, the algorithm is doomed to try to invert \(h_\theta\).

Now we want the identity of \(\theta_{\text{pub}}\) to be revealed on any input to the function \(f_\theta\), but want the behavior of \(f_\theta\) to be essentially that of \(g_\theta\). In order to achieve this, let \(\lfloor \cdot \rfloor\) be the function which truncates a number to \(2d + 2\) bits of precision. This is sufficient precision to distinguish between the two smallest non-zero numbers used in the construction of \(g_\theta\). Also fix an encoding scheme that maps each \(\theta_{\text{pub}}\) to a unique number \([\theta_{\text{pub}}]\). We do this in a manner such that \(2^{-2p} \leq [\theta_{\text{pub}}] < 2^{-p-1}\).

We will define \(f_\theta\) by letting \(f_\theta(x, a) = [g_\theta(x, a)]_* + [\theta_{\text{pub}}]\). Intuitively, \(f_\theta\) mimics the behavior of \(g_\theta\) in its first \(p\) bits, then encodes the identity of \(\theta_{\text{pub}}\) in its subsequent \(p\) bits. \([\theta_{\text{pub}}]\) is the smallest output of \(f_\theta\), and “acts as” \(\theta_{\text{pub}}\).

The subsequent lemma establishes that the first two conditions of Theorem 2 are satisfied by \(F_\theta\).

**Lemma 1.** For any \(f_\theta \in \mathcal{F}_\theta\) and any fixed \(x \in \mathcal{X}\), \(f(x, \cdot)\) can be optimized from a constant number of queries, and poly\((d)\) computation. Furthermore, there exists an efficient algorithm for the supervised no-regret problem on \(\mathcal{F}_\theta\) with err\((T) = O(\log T)\), requiring poly\((d)\) computation per step.

**Proof.** For any \(\theta\), the fixed-state optimization problem on \(f_\theta(x, \cdot)\) is solved by simply querying the special action \(a^*\). If \(f_\theta(x, a^*) < 2^{-p-1}\), then \(g_\theta(x, a^*) = 0\), and \(x\) is not in the image of \(h_\theta\). Therefore, \(a^*\) is a maximizing action, and we are done. Otherwise, \(f_\theta(x, a^*)\) uniquely identifies the optimal action \(h_\theta^{-1}(x)\), which we can subsequently query.

The supervised no-regret problem is similarly trivial. Consider the following algorithm. On the first state, it queries an arbitrary action, extracts its \(p\) lowest order bits, learning \(\theta_{\text{pub}}\). The algorithm can now compute the value of \(f_\theta(x, a)\) on any \((x, a)\) pair where \(a \in \mathbb{Z}_n\). If \(a \in \mathbb{Z}_n\), the algorithm simply checks if \(h_\theta(a) = x\). If so, it outputs \(1 + [\theta_{\text{pub}}]\). Otherwise, it outputs \([\theta_{\text{pub}}]\).

The only inputs on which it might make a mistake take the form \((x, a^*)\). If the algorithm has seen the specific pair \((x, a^*)\), it can simply repeat the previously seen value of \(f_\theta(x, a^*)\), resulting in zero error. Otherwise, if \((x, a^*)\) is a new input, the algorithm outputs \([\theta_{\text{pub}}]\), suffering \(\lfloor \frac{0.5}{1 + h_\theta^{-1}(x)} \rfloor\) error. Hence, after the first round, the algorithm cannot suffer error greater than \(\sum_{t=1}^T \frac{0.5}{t} = O(\log T)\).

Finally, we argue that that an efficient no-regret algorithm for the large-scale bandit problem defined by \((\mathcal{F}_\theta, \mathcal{A}, \mathcal{X})\) can be used as a black box to invert any \(h_\theta \in H_\theta\).

**Lemma 2.** Under standard cryptographic assumptions, there is no polynomial \(q\) and efficient algorithm BANDIT for the large-scale bandit problem on \(\mathcal{F}_\theta\) that guarantees \(\sum_{t=1}^T \max_{a_i} f_\theta(x_t, a_i^*) - f_\theta(x_t, a_i) < .5T\) with probability greater than \(1/2\) when \(T = q(d)\).

**Proof.** Suppose that there were such a \(q\), and algorithm BANDIT.

We can design an algorithm that takes \(\theta_{\text{pub}}\) and \(h_\theta(k^*)\) as input, for some unknown \(k^*\) chosen uniformly at random, and outputs an \(a \in \mathbb{Z}_n\) such that \(P(h_\theta(k) = h_\theta(a)) \geq \frac{1}{2q(d)}\).

Consider simulating BANDIT for \(T\) rounds. On each round \(t\), the state provided to BANDIT will be generated by selecting an action \(k_t\) from \(\mathbb{Z}_n\) uniformly at random,
and then providing BANDIT with the state \( h_\theta(k_i) \). At which point, BANDIT will output an action and demand a reward. If the action selected by bandit is the special action \( a^* \), then its reward is simply \( \lfloor 0.5/(1+k) \rfloor_s + |\theta_{pub}| \). If the action selected by bandit is \( a^t \) satisfying \( h_\theta(a^t) = h_\theta(k_i) \), its reward is \( 1 + |\theta_{pub}| \). Otherwise, its reward is \( |\theta_{pub}| \).

By hypothesis, with probability \( 1/2 \), the actions \( a^t \) generated by BANDIT must satisfy \( h_\theta(a^t) = h_\theta(k_i) \) for at least one round \( t \leq T \). Thus, if we choose a round \( \tau \) uniformly at random from \( \{1, \ldots, q(T)\} \), and give state \( h_\theta(k^*) \) to BANDIT on that round, the action \( a^\tau \) returned by bandit will satisfy \( P(h_\theta(a^\tau) = h_\theta(k^*)) \geq \frac{1}{2q(d)} \). This inverts \( h_\theta(k^*) \), and contradicts the assumption that \( h_\theta \) belongs to a family of cryptographic trapdoor functions.

**A.4. Proof of Theorem 5**

We now show that relaxing KWIK to supervised no-regret insufficient to imply no-regret on MAB.

**Restatement of Theorem 5:**

(Relaxing KWIK to supervised no-regret insufficient to imply no-regret on MAB) There exists a class \( \mathcal{F} \) that is supervised no-regret learnable such that if \( N(t) = \sqrt{T} \), for any learning algorithm \( A \) and any \( T \), there is a sequence of trials in the arriving action model such that \( R_A(T)/T > c \) for some constant \( c > 0 \).

**Proof.** First we describe the class \( \mathcal{F} \). For any \( n \)-bit string \( x \), let \( f_x \) be a function such that \( f_x(x) \) is some large value, and for any \( x' \neq x \), \( f_x(x') = 0 \). It’s easy to see that \( \mathcal{F} \) is notKWIK learnable with a polynomial number of don’t-knows — we can keep feeding an algorithm different inputs \( x' \neq x \), and as soon as the algorithm makes a prediction, we can re-select the target function to force a mistake. \( \mathcal{F} \) is no-regret learnable, however: we just keep predicting 0. As soon as we make a mistake, we learn \( x \), and we’ll never err again, so our regret is at most \( O(1/T) \).

Now in the arriving action model, suppose we initially start with \( r \) distinct functions/actions \( f_i = f_{x_i}, i = 1, \ldots, r \). We will choose \( N(T) = \sqrt{T} \), which is sublinear, and \( r = \sqrt{T} \), and we can make \( T \) as large as we want. So we have a no-regret-learnable \( \mathcal{F} \) and a sublinear arrival rate; now we argue that the arriving action MAB problem is hard.

Pick a random permutation of the \( f_i \), and let \( i \) be the indices in that order for convenience. We start the task sequence with all \( x_1 \)'s. The MAB learner faces the problem of figuring out which of the unknown \( f_i \)'s has \( x_1 \) as its high-payoff input. Since the permutation was random, the expected number of assignments of \( x_1 \) to different \( f_i \) before this is learned is \( r/2 \). At that point, all the learner has learned is the identity of \( f_1 \) — the fact that it learned that other \( f_i(x_1) = 0 \) is subsumed by learning \( f_1(x_1) \) is large, since the \( f_i \) are all distinct.

We then continue the sequence with \( x_2 \)'s until the MAB learner identifies \( f_2 \), which now takes \( (r-1)/2 \) assignments in expectation. Continuing in this vein, the expected number of assignments made before learning (say) half of the \( f_i \) is \( \sum_{j=1}^{r/2}(r-j)/2 = \Omega(r^2) = \Omega(T) \). On this sequence of \( \Omega(T) \) tasks, the MAB learner will have gotten non-zero payoff on only \( r = \sqrt{T} \) rounds. The offline optimal, on the other hand, always knows the identity of the \( f_i \) and gets large payoff on every single task. So any learner’s cumulative regret to offline grows linearly with \( T \).