A. Appendix

A.1. Proof of Corollary 1

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

Proof. Let $A(\epsilon, \delta)$ denote Algorithm 1 when parameterized by ϵ and δ. We construct a no-regret algorithm A^* for the MAB problem on $\mathcal{F}_\Theta$ 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 τ_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 γ_i be the average regret suffered on the ith epoch. In other words, if $\mathbf{x}^{i,t}$ ($\mathbf{a}^{i,t}$) is the tth state (action) in the ith epoch, then $\gamma_i = E\left[\frac{1}{\tau_i}\sum_{t=1}^{\tau_i} \max_{\mathbf{a}_*^{i,t}\in\mathcal{A}} f_\theta(\mathbf{x}^{i,t}, \mathbf{a}_*^{i,t}) - f_\theta(\mathbf{x}^{i,t}, \mathbf{a}^{i,t})\right]$. We therefore can express the average regret of A^* as:

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

From Theorem 1, we know there exists a T_i and choices for ϵ_i and δ_i so that $\gamma_i < 2^{-i}$ so long as $\tau_i \geq T_i$. Let $\tau_1 = T_1$, and $\tau_i = \max\{2\tau_{i-1}, T_i\}$. These choices for τ_i, ϵ_i and δ_i 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:

$$\begin{aligned} R_{A^*}(T)/T &\leq \frac{1}{T}\sum_{i=1}^{e(T)} 2^{-(e(T)-i)}\tau_{e(T)}\gamma_i \\ &< \frac{1}{T}\sum_{t=1}^{e(T)} 2^{-e(T)}\tau_{e(T)} \leq e(T)2^{-e(T)} \end{aligned}$$

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 $\mathbf{B}(\epsilon, \delta) = O(\epsilon^{-d}\log^k \delta^{-1})$ for some $d > 0, k \geq 0$ then there are choices of ϵ, δ so that the average regret of Algorithm 1 is

$$O\left(\left(\frac{1}{T}\right)^{\frac{1}{d+1}} \log^k T\right)$$

Proof. Taking $\epsilon = \left(\frac{1}{T}\right)^{\frac{1}{d+1}}$ 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 $\mathcal{F}_\theta$ satisfying the conditions of the theorem.

Restatement of Theorem 2:
There exists a class of models $\mathcal{F}_\theta$ such that

- *$\mathcal{F}_\Theta$ is fixed-state optimizable;*
- *There is an efficient algorithm A such that on an arbitrary sequence of T trials $\mathbf{z}^t$, A makes a prediction $\hat{y}^t$ of $y^t = f_\theta(\mathbf{z}^t)$ and then receives y^t as feedback, and the total regret $\sum_{t=1}^T |y^t - \hat{y}^t|$ is sublinear 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 $\mathcal{F}_\Theta$.*

Let $\mathbb{Z}_n = \{0, ..., n-1\}$. Suppose that Θ parameterizes a family of cryptographic trapdoor functions H_Θ (which we will use to construct $\mathcal{F}_\theta$). Specifically, each θ consists of a "public" and "private" part so that $\theta = (\theta_{\text{pub}}, \theta_{\text{pri}})$, and $H_\Theta = \{h_\theta : \mathbb{Z}_n \to \mathbb{Z}_n\}$. The cryptographic guarantee ensured by H_Θ is summarized in the following definition.

Definition 1. *Let $d = \lceil \log |\mathbb{Z}_n| \rceil$. Any family of cryptographic trapdoor functions H_Θ must satisfy the following conditions:*

- *(Efficiently Computable) For any θ, knowing just θ_{pub} gives an efficient (polynomial in d) algorithm for computing $h_\theta(\mathbf{a})$ for any $\mathbf{a} \in \mathbb{Z}_n$.*
- *(Not Invertible) Let k be chosen uniformly at random from $\mathbb{Z}_n$. Let A be an efficient (randomized) algorithm that takes θ_{pub} and $h_\theta(k)$ as input (but not θ_{pri}), and outputs an $\mathbf{a} \in \mathbb{Z}_n$. There is no polynomial q such that $P(h_\theta(k) = h_\theta(\mathbf{a})) \geq 1/q(d)$.*

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 $\mathcal{A} = \mathbb{Z}_n \cup \{\mathbf{a}^*\}$. For any $h_\theta \in H_\theta$, let h_θ^{-1} denote the inverse function to h_θ. Since h_θ may be many-to-one, for any y in the image of h_θ, 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_Θ. The behavior of each g_θ will be essentially identical to that of f_θ, and for the purposes of understanding the construction, it is useful to think of them as being exactly identical.

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

On action $\mathbf{a}^*$, $g_\theta(\mathbf{x}, \mathbf{a}^*)$ reveals the location of $h_\theta^{-1}(\mathbf{x})$. Specifically $g_\theta(\mathbf{x}, \mathbf{a}^*) = \frac{0.5}{1+h_\theta^{-1}(\mathbf{x})}$ if $\mathbf{x}$ has an inverse and $g_\theta(\mathbf{x}, \mathbf{a}^*) = 0$ if $\mathbf{x}$ is not in the image of h_θ.

It's useful to pause here, and consider the purpose of the construction. Assume that θ_{pub} is known. Then if $\mathbf{x}$ and $\mathbf{a}$ ($\mathbf{a} \in \mathbb{Z}_n$) are presented simultaneously in the supervised learning setting, it's easy to simply check if $h_\theta(\mathbf{x}) = \mathbf{a}$, making accurate predictions. In the fixed-state optimization setting, querying $\mathbf{a}^*$ presents the algorithm with all the information it needs to find a maximizing action. However, in the bandit setting, if a new $\mathbf{x}$ is being drawn uniformly at random and presented to the algorithm, the algorithm is doomed to try to invert h_θ.

Now we want the identity of θ_{pub} to be revealed on any input to the function f_θ, but want the behavior of f_θ to be essentially that of g_θ. In order to achieve this, let $\lfloor \cdot \rfloor_*$ be the function which truncates a number to $p = 2d + 2$ bits of precision. This is sufficient precision to distinguish between the two smallest non-zero numbers used in the construction of g_θ, $\frac{1}{2}\frac{1}{n}$ and $\frac{1}{2}\frac{1}{n-1}$. Also fix an encoding scheme that maps each θ_{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_θ by letting $f_\theta(\mathbf{x}, \mathbf{a}) = \lfloor g_\theta(\mathbf{x}, \mathbf{a}) \rfloor_* + [\theta_{\text{pub}}]$. Intuitively, f_θ mimics the behavior of g_θ in its first p bits, then encodes the identity of θ_{pub} in its subsequent p bits. $[\theta_{\text{pub}}]$ is the smallest output of f_θ, and "acts as" zero.

The subsequent lemma establishes that the first two conditions of Theorem 2 are satisfies by F_Θ.

Lemma 1. *For any $f_\theta \in \mathcal{F}_\Theta$ and any fixed $\mathbf{x} \in \mathcal{X}$, $f(\mathbf{x}, \cdot)$ 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 θ, the fixed-state optimization problem on $f_\theta(\mathbf{x}, \cdot)$ is solved by simply querying the special action $\mathbf{a}^*$. If $f_\theta(\mathbf{x}, \mathbf{a}^*) < 2^{-p-1}$, then $g_\theta(\mathbf{x}, \mathbf{a}^*) = 0$, and $\mathbf{x}$ is not in the image of h_θ. Therefore, $\mathbf{a}^*$ is a maximizing action, and we are done. Otherwise, $f_\theta(\mathbf{x}, \mathbf{a}^*)$ uniquely identifies the optimal action $h^{-1}(\mathbf{x})$, 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 θ_{pub}. The algorithm can now compute the value of $f_\theta(\mathbf{x}, \mathbf{a})$ on any $(\mathbf{x}, \mathbf{a})$ pair where $\mathbf{a} \in \mathbb{Z}_n$. If $\mathbf{a} \in \mathbb{Z}_n$, the algorithm simply checks if $h_\theta(\mathbf{a}) = \mathbf{x}$. 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 $(\mathbf{x}, \mathbf{a}^*)$. If the algorithm has seen the specific pair $(\mathbf{x}, \mathbf{a}^*)$, it can simply repeat the previously seen value of $f_\theta(\mathbf{x}, \mathbf{a}^*)$, resulting in zero error. Otherwise, if $(\mathbf{x}, \mathbf{a}^*)$ is a new input, the algorithm outputs $[\theta_{\text{pub}}]$, suffering $\lfloor \frac{0.5}{1+h^{-1}(\mathbf{x})} \rfloor_*$ 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 $\sum_{t=1}^{T} \max_{\mathbf{a}_*^t} f_\theta(\mathbf{x}_t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}_t, \mathbf{a}^t) < .5T$ with probability greater than $1/2$ when $T \leq q(d)$.*

Proof. Suppose that there were such a q, and algorithm `BANDIT`.

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

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 $h_\theta(k_t)$. At which point, `BANDIT` will output an action and demand a reward. If the action selected by bandit is the special action $\mathbf{a}^*$, then its reward is simply $\lfloor 0.5/(1+k) \rfloor_* + [\theta_{\text{pub}}]$. If the action selected by bandit is $\mathbf{a}^t$ satisfying $h_\theta(\mathbf{a}^t) = h_\theta(k)$, its reward is $1 + [\theta_{\text{pub}}]$. Otherwise, it's reward is $[\theta_{\text{pub}}]$.

By hypothesis, with probability $1/2$, the actions $\mathbf{a}^t$ generated by `BANDIT` must satisfy $h(\mathbf{a}^t) = h_\theta(k_t)$ for at least one round $t \leq T$. Thus, if we choose a round τ uniformly at random from $\{1, ..., q(T)\}$, and give state $h_\theta(k^*)$ to `BANDIT` on that round, the action $\mathbf{a}^\tau$ returned by bandit will satisfy $P(h_\theta(\mathbf{a}^\tau) = h_\theta(k)) \geq \frac{1}{2q(d)}$. This inverts $h_\theta(k^*)$, and contradicts the assumption that h_θ 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 not KWIK 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 $f_i = f_{x_i} \in \mathcal{F}$, $i = 1, \ldots, r$. 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_is 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 identify 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. □