A. Proofs

Lemma 1 (Ross & Bagnell Lemma 4.3)**.** *For any two policies,* π_1, π_2*:*

$$J(\pi_1) - J(\pi_2) = T\mathbb{E}_{t\sim U(1,T), s\sim d^t_{\pi_1}}\left[Q^{\pi_2}(s,\pi_1) - Q^{\pi_2}(s,\pi_2)\right] = T\mathbb{E}_{t\sim U(1,T), s\sim d^t_{\pi_2}}\left[Q^{\pi_1}(s,\pi_1) - Q^{\pi_1}(s,\pi_2)\right]$$

Proof. Let π^t be a policy that executes π_1 in the first t steps and then executes π_2 from time steps $t+1$ to T. We have $J(\pi_1) = J(\pi^T)$ and $J(\pi_2) = J(\pi^0)$. Consequently, we can set up the telescoping sum:

$$\begin{aligned} J(\pi_1) - J(\pi_2) &= \sum_{t=1}^{T}\left[J(\pi^t) - J(\pi^{t-1})\right] = \sum_{t=1}^{T}\left[\mathbb{E}_{s\sim d^t_{\pi_1}}\left[Q^{\pi_2}(s_t,\pi_1) - Q^{\pi_2}(s_t,\pi_2)\right]\right] \\ &= T\mathbb{E}_{t\sim U(1,T), s\sim\pi_1}\left[Q^{\pi_2}(s,\pi_1) - Q^{\pi_2}(s,\pi_2)\right] \end{aligned}$$

The second equality in the lemma can be obtained by reversing the roles of π_1 and π_2 above. □

A.1. Proof of Theorem 3

We start with an application of Lemma 1. Using the lemma, we have:

$$\begin{aligned} J(\bar{\pi}) - J(\pi^{\text{ref}}) &= \frac{1}{N}\sum_i\left[J(\hat{\pi}_i) - J(\pi^{\text{ref}})\right] \\ &= \frac{1}{N}\sum_i\left[T\mathbb{E}_{t\sim U(1,T), s\sim\hat{\pi}_i}\left[Q^{\pi^{\text{ref}}}(s,\hat{\pi}_i) - Q^{\pi^{\text{ref}}}(s,\pi^{\text{ref}})\right]\right] \end{aligned} \tag{6}$$

We also observe that

$$\begin{aligned} &\sum_{t=1}^{T}\left(J(\bar{\pi}) - \min_{\pi\in\Pi}\mathbb{E}_{s\sim d^t_{\bar{\pi}}}[Q^{\bar{\pi}}(s,\pi)]\right) \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[J(\hat{\pi}_i) - \min_{\pi\in\Pi}\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}[Q^{\hat{\pi}_i}(s,\pi)]\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}[Q^{\hat{\pi}_i}(s,\hat{\pi}_i)] - \min_{\pi\in\Pi}\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}[Q^{\hat{\pi}_i}(s,\pi)]\right] \\ &\le \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}[Q^{\hat{\pi}_i}(s,\hat{\pi}_i) - \min_a Q^{\hat{\pi}_i}(s,a)]\right]. \end{aligned} \tag{7}$$

Combining the above bounds from Equations 6 and 7, we see that

$$\begin{aligned} &\beta\left(J(\bar{\pi}) - J(\pi^{\text{ref}})\right) + (1-\beta)\sum_{t=1}^{T}\left(J(\bar{\pi}) - \min_{\pi\in\Pi}\mathbb{E}_{s\sim d^t_{\bar{\pi}}}[Q^{\bar{\pi}}(s,\pi)]\right) \\ \le&\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}\left[\beta\left(Q^{\pi^{\text{ref}}}(s,\hat{\pi}_i) - Q^{\pi^{\text{ref}}}(s,\pi^{\text{ref}})\right) + (1-\beta)\left(Q^{\hat{\pi}_i}(s,\hat{\pi}_i) - \min_a Q^{\hat{\pi}_i}(s,a)\right)\right] \\ =&\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}\left[Q^{\pi_i^{\text{out}}}(s,\hat{\pi}_i) - \beta Q^{\pi^{\text{ref}}}(s,\pi^{\text{ref}}) - (1-\beta)\min_a Q^{\hat{\pi}_i}(s,a)\right] \\ \le&\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}\left[Q^{\pi_i^{\text{out}}}(s,\hat{\pi}_i) - \beta\min_a Q^{\pi^{\text{ref}}}(s,a) - (1-\beta)\min_a Q^{\hat{\pi}_i}(s,a)\right] \end{aligned}$$

A.2. Proof of Corollary 1

The proof is fairly straightforward from definitions. By definition of no-regret, it is immediate that the gap

$$\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}\left[c_{i,t}(\hat{\pi}_i(s_t)) - c_{i,t}(\pi(s_t))\right] = o(NT), \tag{8}$$

for all policies $\pi \in \Pi$, where we recall that $c_{i,t}$ is the cost-vector over the actions on round i when we do roll-outs from the t_{th} decision point. Let $\mathbb{E}_i$ denote the conditional expectation on round i, conditioned on the previous rounds in Algorithm 1. Then it is easily seen that

$$\mathbb{E}_i[c_{i,t}(a)] = \mathbb{E}_i\left[\ell(e_{i,t}(a)) - \min_{a'}\ell(e_{i,t}(a'))\right],$$

with $e_{i,t}$ being the end-state reached on completing the roll-out with the policy π_i^{out} on round i, when action a was taken on the decision point t. Recalling that we rolled in following the trajectory of π_i^{in}, this expectation further simplifies to

$$\mathbb{E}_i[c_{i,t}(a)] = \mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}\left[Q^{\pi_i^{\text{out}}}(s,a)\right] - \mathbb{E}_i\left[\min_{a'}\ell(e_{i,t}(a'))\right].$$

Now taking expectations in Equation 8 and combining with the above observation, we obtain that for any policy $\pi \in \Pi$,

$$\begin{aligned}&\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}\left[c_{i,t}(\hat{\pi}_i(s_t)) - c_{i,t}(\pi(s_t))\right]\\ &= \sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}_{s\sim d^t_{\hat{\pi}_i}}\left[Q^{\pi_i^{\text{out}}}(s,\hat{\pi}_i(s)) - Q^{\pi_i^{\text{out}}}(s,\pi(s))\right] = o(NT).\end{aligned}$$

Taking the best policy $\pi \in \Pi$ and dividing through by NT completes the proof.

A.3. Proof sketch of Theorem 5

(Sketch only) We decompose the analysis over exploration and exploitation rounds. For the exploration rounds, we bound the regret by its maximum possible value of 1. To control the regret on the exploitation rounds, we focus on the updates performed during exploration.

The cost vector $\hat{c}(a)$ used at an exploration round i satisfies

$$\begin{aligned}\mathbb{E}_i[\hat{c}(a)] &= \mathbb{E}_i\left[K\ell(e(a_t))\mathbf{1}[a=a_t]\right]\\ &= \mathbb{E}_{t\sim U(0:T-1),s\sim d^t_{\hat{\pi}_{n_i}}}\left[Q^{\pi_i^{\text{out}}}(s,a)\right],\end{aligned}$$

Corollary 1. Since the cost vector is identical in expectation as that used in Algorithm 1, the proof of theorem 3, which only depends on expectations, can be reused to prove a result similar to theorem 3 for the exploration rounds. That is, letting $\bar{\pi}_i$ to be the averaged policy over all the policies in $\mathcal{I}$ at exploration round i, we have the bound

$$\begin{aligned}\beta(J(\bar{\pi}_i) - J(\pi^{\text{ref}})) + (1-\beta)\sum_{t=1}^{T}\left(J(\bar{\pi}) - \min_{\pi\in\Pi}\mathbb{E}_{s\sim d^t_{\bar{\pi}}}[Q^{\bar{\pi}}(s,\pi)]\right)\\ \leq T\delta_i,\end{aligned}$$

where δ_i is as defined in Equation 4.

On the exploitation rounds, we can now invoke this guarantee. Recalling that we have n_i exploration rounds until round i, the expected regret at an exploitation round i is at most δ_{n_i}. Thus the overall regret of the algorithm is at most

$$\text{Regret} \leq \epsilon + \frac{1}{N}\sum_{i=1}^{N}\delta_{n_i},$$

which completes the proof.

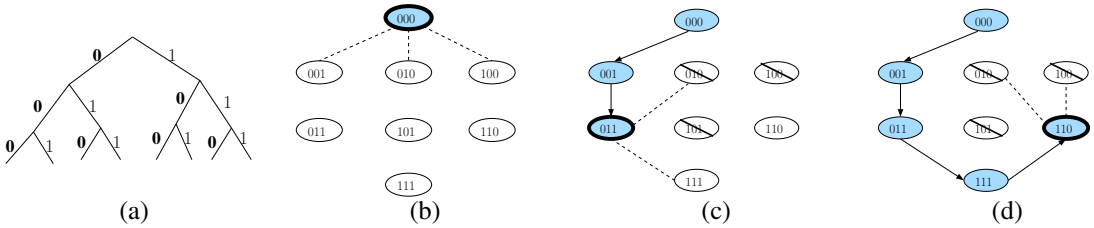

Figure 4. Pictorial illustration of the proof elements of Theorem 4. Panel (a) depicts the actions chosen by policy 000. Selected action in each state is indicated in bold. Panels (b) through (d) depict various stages as the algorithm updates the policy to its one-step deviations, starting from the policy 000. Each policy that the algorithm selects is depicted by a shaded circle, with the arrows marking the moves of the algorithm. Current policy is the shaded circle with a bold boundary. Dashed lines denote the potential one-step deviations that the algorithm can move to and crossed policies are those which have higher costs than the current policy (see text for details).

A.4. Proof of corollary 2

We start by substituting Equation 5 in the regret bound of Theorem 5. This yields

$$\text{Regret} \leq \epsilon + \frac{T}{N}\delta_{\text{class}} + \frac{cKT}{N}\sum_{i=1}^{N}\sqrt{\frac{\log|\Pi|}{n_i}}.$$

We would like to further replace n_i with its expectation which is ϵi. However, this does not yield a valid upper bound directly. Instead, we apply a Chernoff bound to the quantity n_i, which is a sum of i i.i.d. Bernoulli random variables with mean ϵ. Consequently, we have

$$\mathbb{P}\left(n_i \leq (1-\gamma)\epsilon i\right) \leq \exp\left(-\frac{\gamma^2\epsilon i}{2}\right) \leq \exp(-\epsilon i/8),$$

for $\gamma = 1/2$. Let $i_0 = 16\log N/\epsilon + 1$. Then we can sum the failure probabilities above for all $i \geq i_0$ and obtain

$$\begin{aligned}\sum_{i=i_0}^{N}\mathbb{P}\left(n_i \leq \epsilon i/2\right) &\leq \sum_{i=i_0}^{N}\exp(-\epsilon i/8) \leq \sum_{i=i_0}^{\infty}\exp(-\epsilon i/8)\\ &\leq \frac{\exp(-\epsilon i_0/8)}{1-\exp(-\epsilon/8)}\\ &= \frac{\exp(-2\log N)}{\exp(\epsilon/8)-1} \leq \frac{8}{N^2\epsilon},\end{aligned}$$

where the last inequality uses $1 + x \leq \exp(x)$. Consequently, we can now allow a regret of 1 on the first i_0 rounds, and control the regret on the remaining rounds using $n_i \leq \epsilon i/2$. Doing so, we see that with probability at least $1 - 2/(N^2\epsilon)$

$$\begin{aligned}\text{Regret} &\leq \epsilon + \frac{i_0}{N} + \frac{T}{N}\delta_{\text{class}} + \frac{cKT}{N}\sum_{i=1}^{N}\sqrt{\frac{2\log|\Pi|}{\epsilon i}}\\ &\leq \epsilon + \frac{16\log N + \epsilon}{N\epsilon} + \frac{T}{N}\delta_{\text{class}} + \frac{8cKT\log|\Pi|}{\epsilon N}\end{aligned}$$

Choosing $\epsilon = (KT)^{2/3}(\log(N|\Pi|)/N)^{1/3}$ completes the proof.

A.5. Proof of Theorem 4

The proof follows from results in combinatorics. The dynamics of algorithms considered here can be thought of as a path through a graph where the vertices are the corners of the boolean hypercube in T dimensions with two vertices at Hamming distance 1 sharing an edge. We demonstrate that there is a cost function such that the algorithm is forced to traverse a long path before reaching a local optimum. Without loss of generality, assume that the algorithm always moves to a one-step deviation with the lowest cost since otherwise longer paths exist.

Algorithm 3 Cost-sensitive One Against All (CSOAA) Algorithm

Require: Initial predictor $f_1(x)$
1: **for all** $t = 1, 2, \ldots T$ **do**
2: Observe $\{x_{t,i}\}_{i=1}^K$.
3: Predict class $i_t = \arg\min_{i=1}^K f_t(x_{t,i})$.
4: Observe costs $\{c_{t,i}\}_{i=1}^K$.
5: Update f_t using online least-squares regression on data $\{x_{t,i}, c_{t,i}\}_{i=1}^K$.
6: **end for**

To gain some intuition, first consider $T = 3$ which is depicted in Figure 4. Suppose the algorithm starts from the policy 000 then moves to the policy 001. If the algorithm picks the best amongst the one-step deviations, we know that $J(001) \leq \min\{J(000), J(010), J(100)\}$, placing constraints on the costs of these policies which force the algorithm to not visit any of these policies later. Similarly, if the algorithm moves to the policy 011 next, we obtain a further constraint $J(011) < \min\{J(101), J(001)\}$. It is easy to check that the only feasible move (corresponding to policies not crossed in Figure 4(c)) which decreases the cost under these constraints is to the policy 111 and then 110, at which point the algorithm attains local optimality since no more moves that decrease the cost are possible. In general, at any step i of the path, the policy $\hat{\pi}_i$ is a one-step deviation of $\hat{\pi}_{i-1}$ and at least 2 or more steps away from $\hat{\pi}_j$ for $j < i - 1$. The policy never moves to a neighbor of an ancestor (excluding the immediate parent) in the path.

This property is the key element to understand more generally. Suppose we have a current path $\hat{\pi}_1 \to \hat{\pi}_2 \ldots \to \hat{\pi}_{i-1} \to \hat{\pi}_i$. Since we picked the best neighbor of $\hat{\pi}_j$ as $\hat{\pi}_{j+1}$, $\hat{\pi}_{i+1}$ cannot be a neighbor of any $\hat{\pi}_j$ for $j < i$. Consequently, the maximum number of updates the algorithm must make is given by the length of the longest such path on a hypercube, where each vertex (other than start and end) neighbors exactly two other vertices on the path. This is called the *snake-in-the-box* problem in combinatorics, and arises in the study of error correcting codes. It is shown by Abbott & Katchalski (1988) that the length of longest such path is $\Theta(2^T)$. With monotonically decreasing costs for policies in the path and maximal cost for all policies not in the path, the traversal time is $\Theta(2^T)$.

Finally, it might appear that Algorithm 1 is capable of moving to policies which are not just one-step deviations of the currently learned policy, since it performs updates on "mini-batches" of T cost-sensitive examples. However, on this lower bound instance, Algorithm 1 will be forced to follow one-step deviations only due to the structure of the cost function. For instance, from the policy 000 when we assign maximal cost to policies 010 and 100 in our example, this corresponds to making the cost of taking action 1 on first and second step very large in the induced cost-sensitive problem. Consequently, 001 is the policy which minimizes the cost-sensitive loss even when all the T roll-outs are accumulated, implying the algorithm is forced to traverse the same long path to local optimality.

B. Details of cost-sensitive reduction

In this section we present the details of the reduction to cost-sensitive multiclass classification used in our experimental evaluation. The experiments used the Cost-Sensitive One Against All (CSOAA) classification technique, the pseudocode for which is presented in Algorithm 3. In words, the algorithm takes as input a feature vector $x_{t,i}$ for class i at round t. It then trains a regressor to predict the corresponding costs $c_{t,i}$ given the features. Given a fresh example, the predicted label is the one with the smallest predicted cost. This is a natural extension of the One Against All (OAA) approach for multiclass classification to cost-sensitive settings. Note that this also covers the alternative approach of having a common feature vector, $x_{t,i} \equiv z_t$ for all i and instead training K different cost predictors, one for each class. If $z_t \in \mathbb{R}^d$, one can simply create $x_{t,i} \in \mathbb{R}^{dK}$, with $x_{t,i} = z_t$ in the i_{th} block and zero elsewhere. Learning a common predictor f on x is now representationally equivalent to learning K separate predictors, one for each class.

There is one missing detail in the specification of Algorithm 3, which is the update step. The specifics of this step depend on the form of the function $f(x)$ being used. For instance, if $f(x) = w^T x$, then a simple update rule is to use online ridge regression (see e.g. Section 11.7 in (Cesa-Bianchi & Lugosi, 2006)). Online gradient descent (Zinkevich, 2003) on the squared loss $\sum_{i=1}^K (f(x_{t,i}) - c_{t,i})^2$ is another simple alternative, which can be used more generally. The specific implementation in our experiments uses a more sophisticated variant of online gradient descent with linear functions.

C. Details of Experiments

Our implementation is based on Vowpal Wabbit (VW) version 7.8 (http://hunch.net/~vw/). It is available at https://github.com/KaiWeiChang/vowpal_wabbit/tree/icmlexp. For LOLS, we use flags "–search_rollin", "–search_rollout", "–search_beta" to set the rollin policy, the rollout policy, and β, respectively. We use "–search_interpolation policy –search_passes_per_policy –passes 5" to enable SEARN. The details settings of various VW flags for the three experiments are shown below:

- POS tagging: we use "–search_task sequence –search 45 –holdout_off –affix -2w,+2w –search_neighbor_features -1:w,1:w -b 28"
- Dependency parsing: we use " –search_task dep_parser –search 12 –holdout_off –search_history_length 3 –search_no_caching -b 24 –root_label 8 –num_label 12"
- Cost-sensitive multiclass: we use "–search_task multiclasstask –search 5 –holdout_off –mc_cost"

The data sets used in the experiments are available upon request.