We thank all reviewers for their thoughtful comments.

Experimental setup (R1, R3): The reviewers brought up an excellent suggestion to deploy our algorithm in production on real users, for example using a user-interactive system. We agree this is an appropriate next step for development. However, applying contextual bandits to continuous-action domains is a relatively new approach; the large-scale contextual bandit datasets for the examples in the introduction have not been collected yet.

Simulating contextual bandit learning using regression/classification datasets is a standard protocol for academic evaluation of bandit algorithms [see e.g. A. Bietti et al. A Contextual Bandit Bake-off]. Note that we have chosen the regression datasets with the criterion of having million scale examples with unique regression values, so it is very relevant to a real contextual bandit setting.

Approximate ERM guarantee of `Train_tree` (R2, R4): At a high level, our analysis of the `Train_tree` algorithm needs to account for compounding errors accumulated in each of the nodes, which results in a $O(K\sqrt{1/n})$ rate slower than the $O(\sqrt{K/n})$ rate achieved by ERM. We will clarify this in the final version.

Realizability assumption (R3, R4): Indeed we assume realizability in Theorems 6 and 7. This is a typical situation in ML theory when the theory does not necessarily capture *all* cases when things work, but instead gives justification and assurance that they do work in practice.

Realizability has been widely used in many successful contextual bandit algorithms (e.g. LinUCB, OFUL).

Regarding checking realizability: This argument applies to essentially all supervised learning settings, where standard surrogate loss minimization can fail without realizability or related unverifiable assumptions.

We now answer specific questions raised by individual reviewers:

R1: We agree that in our online contextual bandit learning setting, tuning hyperparameters h and K can be problematic. Note that all other hyperparameters such as learning rate, the greedy parameter, and the penalty term in line 5 of Algorithm 3 (`CATS_Off`) are fixed in our experiments and for all of the datasets. For practical applications, we recommend using `CATS_Off` to perform off-policy optimization to find the best (h, K) combination, whose effectiveness is demonstrated in Figure 1 (right) in the paper. Note that in `CATS_Off`, we only use input logged data to select the best h, K hyperparameters.

Regarding the choice of h and K in bias-variance trade-off: Tuning K has the usual bias-complexity trade-off in standard statistical learning. If we are competing with the best (non-smoothed) policy in a class, then tuning h also incurs a bias-variance tradeoff: (1) the smoothed loss more closely approximates the true loss for smaller h; (2) a smaller h makes achieving a low h-smoothed regret bound harder.

R2: The reviewer's understanding regarding the `dTree` baseline and the filter tree algorithm is correct. We will clarify this in the final version. We will elaborate more on line 208-213 in the final version.

R3: The reviewer mentioned our accuracy is similar to the `dLinear` baseline. This is true, but our algorithm beats `dLinear` in terms of time complexity. As R4 has acknowledged, our algorithm wins over `dLinear` computationally and wins over `dTree` statistically.

The reviewer has suggested three papers to compare our algorithm with. Note that [1] was published *after* the submission deadline so we could not have possibly compared to it. [2] and [3] are also not directly relevant to our work since (1) they impose structure on contexts, rather than actions, and moreover the structure is *linear* rather than Lipschitz; (2) they work with discrete action spaces. We thank the reviewer for bringing these papers to our attention and will cite them in our final draft.

We will polish the writing on the sections that the reviewer highlighted in the final version.

R4: The idea of smoothing was initially proposed in [35], but their proposed algorithm is computationally intractable. Our contribution is proposing a computationally efficient algorithm for continuous action spaces in the smoothing framework. Our online algorithm `CATS`, its off-policy optimization version `CATS_Off`, and their statistical guarantees as well as our careful implementation design with $O(\log K)$ are all new and constitute the contributions of this work.

See Figure 1 of [35] for an example of a discontinuous loss function, where the best single action is impossible to find within a finite number of rounds of contextual bandit learning.

Regarding the reason we need to guess width but the location can be automatically adjusted: If the learner sets a high discretization level K then it can take actions in [0,1] with fine granularity.