Experiments with Optimal Behavior

Proposed Training and Tuning Strategy for Expert-Generated Data

For datasets where the behavioral policy is already near-optimal, we recommend *behavioral cloning (BC)* as the training algorithm. BC either trains a stochastic policy to minimize negative log-likelihood (NLL) or a deterministic policy to minimize mean squared error (MSE). While both strategies can be advantageous in different scenarios, we focus on the deterministic case here. As BC uses a supervised learning algorithm, its hyperparameters can be tuned in principle to minimize error on a validation set. Theoretical bounds on policy performance can be obtained by controlling the associated generalization error (Ross 2011). Moreover, this offline hyperparm tuning (OHT) strategy avoids estimating any additional unknown quantities. In contrast, model-based evaluation requires estimating an unknown transition kernel, FQE requires estimating an unknown value function and the MSBE requires estimating an unknown Bellman backup.

While this strategy seems intuitive, the question of how well BC performs when its hyperparms are tuned offline using validation MSE remains unresolved. For example, many works have concluded that BC performs poorly with long horizons, and that one should use other methods instead, such as inverse RL (Ross 2011, Ho 2016, Yue 2023). This is because of the fact that BC fails to incorporate information on the transition dynamics, which can make it less statistically efficient and more susceptible to distribution shift and compounding errors. However, such works evaluate BC with hyperparms specified a priori. In practice, the hyperparms for BC can be optimally tuned to minimize MSE on a validation set, whereas competitors such as inverse RL cannot tune hyperparms offline as easily. Other works have shown that BC performs well on state-of-the-art (SOTA) offline benchmarks, though not quite as well as traditional RL (Kumar 2022, Emmons 2022). However, these works tune hyperparms for BC and RL using online interactions, making results very optimistic, and again fail to account for the fact that BC is much easier to tune offline than RL. Finally, Emmons 2022 concluded that validation error performs poorly for OHT with imitation learning. However, this work only deals with conditional and filtered BC, and our own results (see the Mountain Car experiments detailed in the previous report) suggest that many of the root causes that make validation NLL fail for conditional BC may not apply to standard BC.

Experiments on Cartpole, Adroit-Expert and Mujoco-Expert

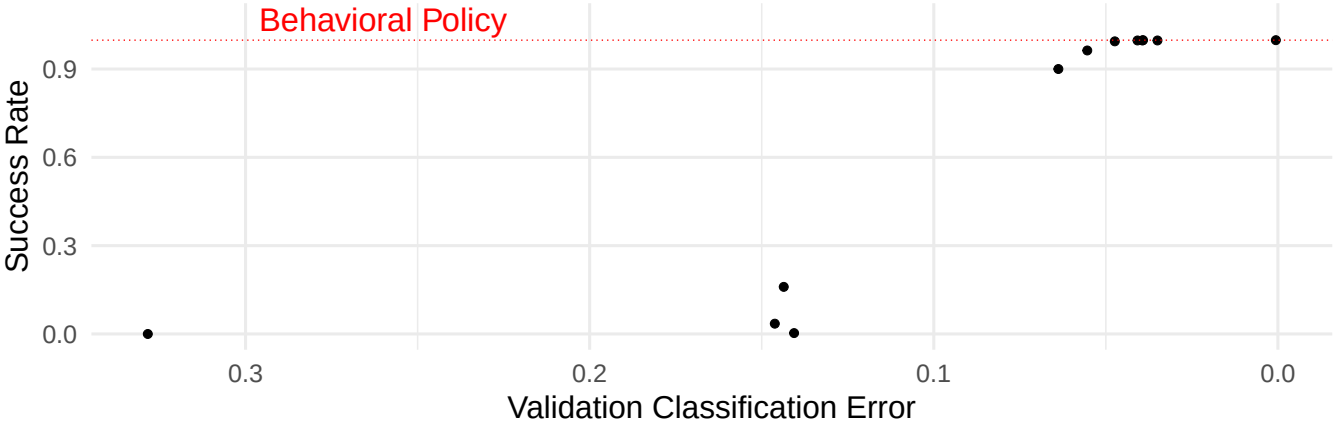

Figure 1: Online Performance vs. Validation Error on Cartpole. BC policies were trained on 40 observed expert trajectories, with 10 trajectories reserved as the validation set. Policies with low validation error consistently perform very well.

We begin by calculating a deterministic expert policy on `cartpole` by applying quadratic FQI to a large offline dataset. This expert policy yields succesful trajectories nearly 100% of the time (success = lasts for 500 steps without terminating). From this expert policy, we generated a small dataset of 50 trajectories. Recalling that actions are binary here, we then generating a dozen BC policies using polynomial ridge logistic regression (with varying polynomial degrees and ridge penalties) and gini index random forests (with varying hyperparms for the minimum node size and number of candidate variables excluded per split), using an 80-20 split for training and validation. We found that BC policies with low validation error consistently performed very well. For example, policies with <5% classification error all achieved comparable performance to the expert policy, whereas those with >10% error did not. Ablation experiments suggested performane was robust to sample sizes, seeds and considered function approximators.

We then considered the datasets `pen-expert-v1` and `walker2d-expert-v2` from the D4RL benchmark. Whereas `cartpole` has four continuous states and a binary action, `walker2d` has 17 continuous states and six continuous actions, and `pen` has 45 continuous states and 25 continuous actions. Both also involve complex dynamics and generate data by following RL-trained policies, with `walker2d-expert` following an SAC-trained policy and `pen-expert` following an NPG-trained policy (both of these RL algorithms estimate *stochastic* policies). The `pen-expert-v1` dataset contains 5000 trajectories corresponding to roughly 500k transitions, with an 80% success rate (success = continues to achieve reward >45 after 100 time steps). We subsampled 10% of the data prior to applying BC, and then used a training-validation split of 80-20, as before.

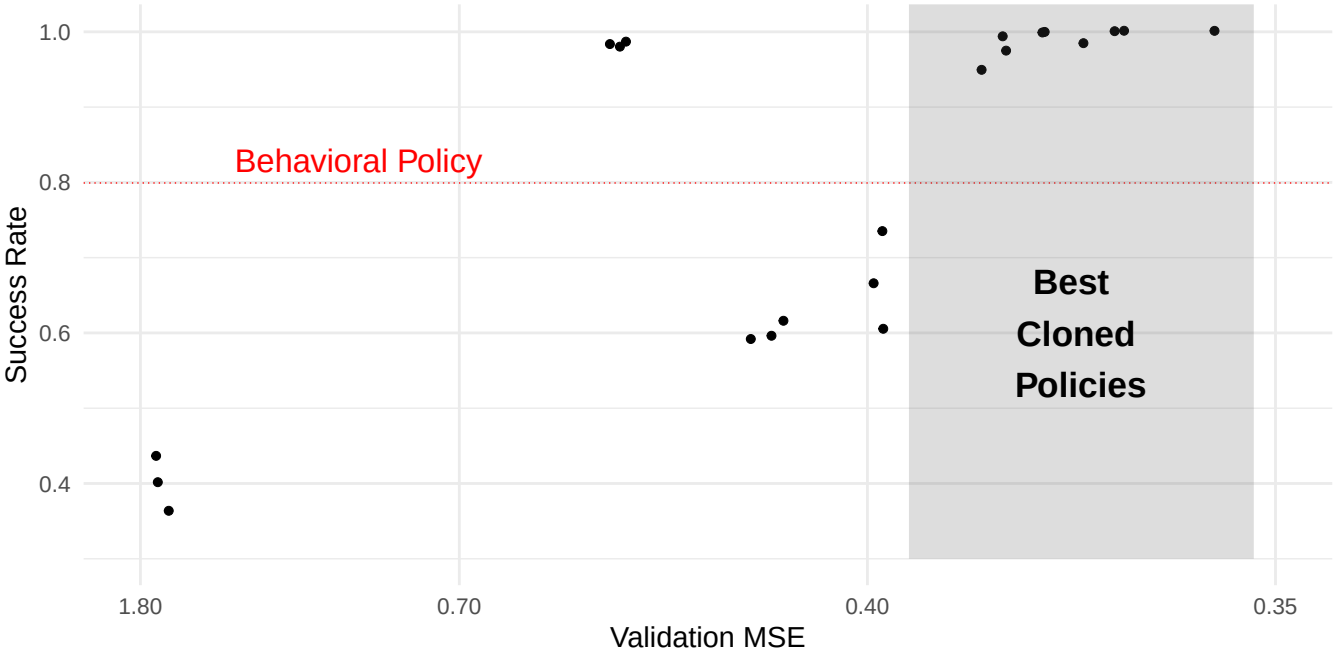

Figure 2: Online Performance vs. Validation Error on Pen-Expert-v1 (10% subsampling). BC Policies were trained on 400 observed expert trajectories, with 100 trajectories reserved as the validation set. All BC policies with sufficiently low validation error perform better than the behavioral policy.

Above we plot success rate vs. validation MSE for a small subset of the policies that were fitted. In the plot we include both networks that severely overfit and severely underfit the training data. We can see that while low validation error is not a neccesary condition for good performance, it is sufficient: all BC policies that have low validation error (those in the grey shaded area) achieve >95% success. This idea of "sufficient but not neccesary" is consistent with previous works that have evaluated other offline metrics such as Bellman errors. Also notice that the best-performing policies actually beat the behavioral policy. As we train BC to minimize MSE, our trained agent is actually executing the actions that the behavioral policy takes *on average*. This can give a denoising effect which filters out mistakes, and shows that even if observed actions are suboptimal, BC can still be of high-quality provided observed decisions are at least high-quality on average.

We also applied BC on the `pen-human-v1` dataset with 1% subsampling, which corresponds to 5000 transitions (50 observed trajectories). While the relationship between validation error and online performance was weaker, tuning

hyperparms to minimize validation error still yielded a policy with a near 100% success rate. Finally, we applied BC to the `walker2d-expert-v1` dataset with 1% subsampling, which corresponds to 10k transitions or a mere 10 trajectories. Unlike previous problems, the Walker2D environment does not have a clear definition of success, and thus we evaluated quality of a policy based on its mean undiscounted return. Once more, we found that low validation MSE was a sufficient condition to ensure excellent performance: while the observed trajectories had a mean return of 4919, a cloned policy fine-tuned with validation error yielded a mean return of nearly 5000. We note that this performance is significantly better than the reported BC performance on existing benchmarks (Yue 2023, Fu 2020). These benchmarks used a fixed, a priori specified architecture for BC, without bothering to conduct any tuning. This shows the importance and utility of OHT.

Experiments on Adroit-Human

From the previous results, one would hope that minimizing validation error always gives performance that rivals or exceeds the behavioral policy. Unfortuantely, this is not the case on all datasets, such as the `pen-human-v1` dataset from D4RL. We discuss our results in more detail followed by a discussion of why performance was subpar for this particular dataset.

`pen-human-v1` which consists of 25 observed trajectories generated by human demonstrations and follows the same MDP as `pen-expert-v1`. We use as a baseline the BC architecture from Kumar 2022, which consists of four 256-unit hidden ReLU layers and a dropout layer (w/ dropout rate=0.2) following every hidden layer. They trained a stochastic policy using NLL, and we modified this training configuration by using an MSE loss function. When trained with early stopping, a default Nadam optimizer and a priori standardization to the observations, the model achieved a validation RMSE of 0.27 and a success rate of 41%. We then manually fine-tuned the architecture and training configuration across various dimensions to try to achieve the smallest RMSE possible, such as its:

- **State representation**, including whether previous observations, previous actions and the time index are used to predict the current action, as opposed to just the most recent observation
- **Hidden network architecture**, including the number of hidden layers, their width and their activation function (ReLU/LeakyReLU/ELU); wide & deep and residual connections; output layer activation (tanh/linear)
- **Optimization strategy**, including the optimization algorithm (NAdam, RMSProp, SGD + Nesterov momentum), learning rate, scheduler, batch size and weight initializer
- **Regularization strategy**, including dropout, batch norm, layer norm and Gaussian noise layers
- (for deterministic policies) **Loss function** (e.g. MSE, MAE, Huber)
- (for stochastic policies) **Distributional family**, including the parametric distributional model (MVN/GMM) and covariance structure (e.g. diagonal, diagonal + low-rank, full, exp/softplus link function)

The final policy uses a wide & deep network, where the deep path consists of three, 600-unit, ELU activation functions initialized using the He uniform initializer, with a dropout layer (dropout rate=0.2) following every hidden layer. Moreover, the inputs are first passed through a normalization layer to have mean zero and unit variance column-wise, followed by a Gaussian noise layer (std=0.01) to improve generalization, while the targets/actions are also standardized column-wise prior to computing the training loss function. The state representation includes the current and previous observation as well as the previous action. The model is trained using a Huber loss function with threshold $\delta = 0.1$. The model is optimized with a NAdam optimizer with learning rate 2e-3, batch size 128, and a performance scheduler that rolls back to the previous best model (according to training loss) and slashes the learning rate by 1/3 whenever training loss does not improve for 10 epochs. Finally, the model is optimized from three random initial values, and we take the result with the lowest validation error as the final model.

Our fine-tuned model achieves a RMSE of 0.07, a 4x improvement in accuracy over the baseline SOTA. Despite this, the deployed cloned policy achieves a lower success rate of 35%. Moreover, other networks with similar accuracy levels had highly variable success rates. For example, using a separate network for the first time point that only uses the most recent observation in the state representation leads to a slightly lower RMSE over the validation set overall, but a success rate of only 27%. Moreover, increasing the Gaussian noise added to the inputs from std=0.01 to std=0.35 lead to a validation RMSE nearly as low as using std=0.01 Gaussian noise, but a success rate of only 20%. All of this suggests that validation error is not performing well here for OHT.

Table 1: Validation Error and Online Performance for a Small Subset of Evaluated Policies on Pen-Human-v1. Even policies with very low validation error (at least as low as we could attain from the given dataset and our tested configurations) have online performance that is highly variable and significantly worse than the behavioral policy.

Policy	Validation.RMSE	Success.Rate
Behavioral	NA	0.56
Kumar (deterministic)	0.273	0.41
Fine-Tuned	0.070	0.35
Fine-Tuned + Special Initial	0.069	0.27
Fine-Tuned + GNoise(std=0.35)	0.076	0.20

One problem with this dataset is that is only consists of 25 trajectories (5000 transitions), which is tiny given the size and complexity of the problem (25 continuous DoF, highly complex and nonlinear dynamics, non-Markovian behavioral policy, long horizon). Because validation MSE only provides a lower bound on online performance, it will be informative only when it is small enough. We can see this from the `pen-expert` plot: among the cloned policies in the grey region, all have consistently good performance. However, if you look outside this region, validation error is no longer strongly associated with performance, and remaining policies can perform very well or very poorly. Similar phenomina have been observed with other offline metrics, like Bellman errors (Zitovsky 2023). This tiny sample siz is likely hindering us from making validation error small enough for it to become a useful metric. Another problem is the quality of behavioral policy. BC can only perform as well as the (average) observed performance, and as the behavioral policy only gives a 56% success rate, observed decisions are clearly suboptimal. Preliminary analysis also suggests that the observed data distribution is extremely narrow. It is well-known that given behavioral policies of similar quality, a data-generating policy that has at least a little diversity in the observed actions will perform much better, as it leads to greater coverage of the state space in the observed data and is thus less susceptible to distribution shift and compounding errors (Ross 2011, Yue 2023). It is perhaps for this reason that BC performs much better on `pen-expert-v1` at 1% subsampling, even though it has the same number of transitions. While `adroit-expert` and `mujoco-expert` datasets were touted in the original D4RL paper as having narrow distribution challenges, our analysis suggests that the distribution of `adroit-human` is even more narrow.

This is not to say that validation error *can't* be effective on such a large and complex problem with such suboptimal observed behavior, but we would likely need more than just 25 trajectories. Fortunately, most real-world imitation datasets have well over 50% success rate (else behavior would not be worthy of imitation), more than 25 trajectories, and/or far less than 25 degrees of freedom. Indeed, our results on Cartpole, `pen-expert` and `walker2d-expert` show that validation error still performs well under reasonable conditions on the size and quality of the dataset which are satisfied for many imitation problems, and thus we feel it is still a useful metric for OHT when the observed behavior is already considered expert-like. And while it is unfortunate that validation error does not perform consistently on all possible datasets, the same can likely be said of all existing metrics proposed for OHT in RL.

There are a few ablation and investigative experiments which we carried out and should briefly mention. First, in addition to training a deterministic policy to minimize validation MSE, we also tried to fit a stochastic policy to minimize validation NLL. Second, to improve sample efficiency we tried to use five-fold cross-validation to estimate validation error. However, instead of refitting the evaluated training algorithm on the full dataset as in traditional 5-fold CV, we used an ensemble of the five models fitted by CV to evaluate online performance. Third, we tried filtering out unsuccessful trajectories. All of these strategies were completely unhelpful in improving online performance or making validation error a useful metric for OHT. Fourth, as BC is known to be susceptable to compounding errors, and as a succesful trajectory is defined as one where the agent is performing desired behavior after 100 time steps, I investigated whether cloned policies performed closer to the behavioral policy when only looking at shorter horizons. However, even when only looking at agent behavior after 50 time steps, performance was still much worse than behavioral policy. Fifth, we ran implicit Q-learning on this dataset, a SOTA RL algorithm whose hyperparms were originally fine-tuned for this specific environment using online interactions. Early stopping was also applied using online interactions. Despite being a SOTA RL algorithm and "cheating" by tuning hyperparms with online interactions, implicit Q-learning could not break 47% success rate. I found BC's performance was similar when online interactions were used to tune its hyperparms.