We also conducted experiments to study the effects of using i.i.d. active learners that ignore the data distribution in RAIL. Figure 2(a) shows the performance of RAIL with three different base active learners, the density-weighted QBC, the standard QBC (without density weighting), and random selection of unlabeled data points. We see that RAIL-DW performs better than both RAIL-QBC and RAIL-RAND. This shows that it is critical for the i.i.d. active learner to exploit the state density information that is estimated by RAIL at each iteration.

**Bicycle Balancing.** Bicycle balancing is a variant of the Bicycle RL benchmark (Randlov and Alstrøm, 1998). The goal is to balance a bicycle moving at a constant speed for 1000 time steps. If the bicycle falls, it remains fallen for the rest of the episode. The state space is described using nine variables measuring various angles, angular velocities, and positions of the bicycle. There are five possible actions each specifying a particular handle-bar torque and rider displacement. The learner’s policy is represented as a linear logistic regression classifier over features of state-action pairs. A feature vector is defined as follows: Given a state $s$, a set of 20 basis functions is computed. This set is repeated for each of the 5 actions giving a feature vector of length 100. The expert policy was hand coded and can balance the bicycle for up to 26K time steps.

We used a similar evaluation procedure as for Cart-pole giving +1 reward for each time step where the bicycle is kept balanced and -1 otherwise. Figure 1(b) compares each approach. The results are similar to those of Cart-pole with RAIL-DW being the top performer. Unif-RAND and Unif-QBC show notably poor performance in this domain. This is because bicycle balancing is a harder learning problem than cart-pole with many more uninformative states (an unrecoverable fall or fallen state). Similarly, 2(b) shows very poor performance for versions of RAIL that use distribution unaware active learners.

**Wargus.** We consider controlling a group of 5 friendly close-range military units against a group of 5 enemy units in the real-time strategy game Wargus, similar to the setup in (Judah et al., 2010). The objective is to win the battle while minimizing the loss in total health of friendly units. The set of actions available to each friendly unit is to attack any one of the remaining units present in the battle (including other friendly units, which is always a bad choice). In our setup, we allow the learner to control one of the units throughout the battle whereas the other friendly units are controlled by a fixed “reasonably good” policy. This situation