

# 15-859(B) Machine Learning Theory

Homework # 6

Solutions

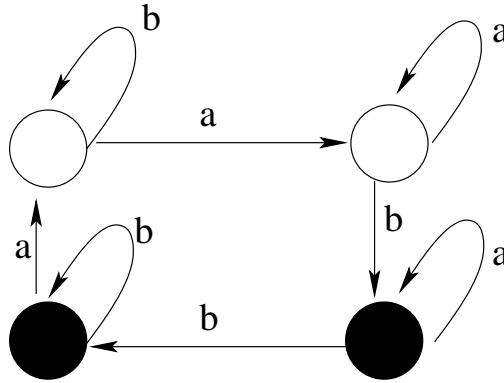
**Exercises:**

1. **DFAs.** A *distinguishing sequence* for a DFA is a sequence of actions such that the observations produced from these actions uniquely determine the starting state. I.e., a sequence  $h$  such that if  $q \neq q'$  then  $\text{obs}(q, h) \neq \text{obs}(q', h)$ . (Here, “ $\text{obs}(q, h)$ ” is the sequence of observations produced by executing  $h$  from  $q$ .)

A *homing sequence* for a DFA is a sequence of actions such that the observations produced from these actions uniquely determine the *ending* state. I.e., a sequence  $h$  such that if  $qh \neq q'h$  then  $\text{obs}(q, h) \neq \text{obs}(q', h)$ . (Here, “ $qh$ ” is the ending state produced by executing  $h$  from  $q$ .)

- (a) Describe a strongly-connected DFA that has no distinguishing sequence. Note that the definition of “ $q \neq q'$ ” is that there must exist a sequence  $h_{qq'}$  such that  $\text{obs}(q, h_{qq'}) \neq \text{obs}(q', h_{qq'})$ , it's just that no single  $h$  works for all pairs.

Solution: Here is one example. Note that any sequence beginning with “a” cannot distinguish the top two states, and any sequence beginning with “b” fails on the bottom two states. This example also has the property that it is strongly-connected.



- (b) Give a homing sequence for your DFA.

Solution: just “a” works. The only possible observation sequences for this action are (white,white), (black,black), or (black,white). They produce ending states of upper-right, lower-right, and upper-left respectively.

2. **Online resource sharing.** Consider a system with  $n$  users and  $m$  resources. User  $i$  has permissions for some subset  $N_i$  of the  $m$  resources (if we construct a bipartite graph with users on the left and resources on the right, then these are the neighbors of user  $i$ ). However, user  $i$  can only use  $k_i \leq |N_i|$  of the  $N_i$  resources at a time. Finally, each resource  $j$  has a size  $s_j$ , and if several users are using a given resource, they have to split it equally. The goal of a user is to maximize total resource usage.

Formally, the game proceeds as follows. Each user  $i$  simultaneously chooses some subset  $\{r_{i_1}, r_{i_2}, \dots, r_{i_{k_i}}\}$  of their  $N_i$  neighbors. Let  $n_j$  be the total number of users who choose resource  $j$ . Then, user  $i$  gets payoff  $\sum_{t=1}^{k_i} s_{i_t} / n_{i_t}$ . (This is equivalent to the market-sharing game of Goemans, Li, Mirrokni and Thottan.)

Suppose we (user  $i$ ) repeatedly play this game each day. We could place this in the framework of “combining expert advice”, except the number of experts  $\binom{|N_i|}{k_i}$  is exponential. Show how you could instead model this in the Kalai-Vempala framework to get a polynomial-time regret-minimizing algorithm. Make sure to argue how you solve the offline problem.

Solution: We have a space of dimension  $N_i$ . The set  $S$  of feasible actions is the set of all vectors in  $\{0, 1\}^{N_i}$  with  $k_i$  1's in it. The cost vector is the vector  $c = (c_1, c_2, \dots, c_{N_i})$  where  $c_t = -s_{it}/(n'_{it} + 1)$  where  $n'_{it}$  is the number of users excluding user  $i$  who chose market  $i_t$ . Thus, the negative of  $x \cdot c$ , where  $x$  is the action chosen by the algorithm, is exactly the algorithm's payoff. We can solve the offline problem by simply taking the sum of all the cost vectors and sorting the entries by cost, choosing the  $k_i$  elements of least cost (highest payoff).

### Problems:

3. **Policy iteration.** The goal of this problem is to prove that policy iteration will eventually reach the optimal policy. Recall that in policy iteration, given some policy  $\pi_i$ , you solve the linear system to compute the state values under that policy:

$$V^{\pi_i}(s) = R(s, \pi_i(s)) + \gamma \sum_{s'} \Pr_{s, \pi_i(s)}(s') V^{\pi_i}(s').$$

(Here, “ $R(s, a)$ ” is the expected reward of executing action  $a$  from state  $s$ .) Then, we define policy  $\pi_{i+1}$  to be the greedy policy with respect to those values. That is,

$$\pi_{i+1}(s) = \arg \max_a \left[ R(s, a) + \gamma \sum_{s'} \Pr_{s, a}(s') V^{\pi_i}(s') \right],$$

and so on.

- (a) As an easy first step, argue that if  $\pi_{i+1} = \pi_i$  (i.e.,  $\pi_{i+1}(s) = \pi_i(s)$  for all states  $s$ ), then  $\pi_i$  is optimal.

Solution: We showed in class that the values  $V(s)$  under the optimal policy are the unique solution to the equations

$$V(s) = \max_a \left[ R(s, a) + \gamma \sum_{s'} \Pr_{s, a}(s') V(s') \right].$$

Thus, if  $\pi(s) = \arg \max_a [R(s, a) + \gamma \sum_{s'} \Pr_{s, a}(s') V^\pi(s')]$  for all  $s$  then  $V^\pi(s)$  satisfies these equations as well and so  $\pi$  must be optimal.

- (b) As the harder second step, argue that the values never decrease (i.e., for all  $s$ ,  $V^{\pi_{i+1}}(s) \geq V^{\pi_i}(s)$ ). This completes the argument because there are only a finite number of different policies.

Hint: what about a hybrid policy that uses  $\pi_{i+1}$  for one step and then  $\pi_i$  from then on? How about  $\pi_{i+1}$  for two steps?

Solution: Let  $\pi_{i+1, k}$  denote the hybrid policy that follows  $\pi_{i+1}$  for  $k$  steps and then follows  $\pi_i$ . Because of the argmax in the definition of  $\pi_{i+1}$  we have that  $V^{\pi_{i+1, 1}}(s) \geq$

$V^{\pi_i}(s)$  for all  $s$ . Now, if the desired theorem were false, then it would have to fail for some finite  $k$  (because the values under the hybrid policy approach that of  $\pi_{i+1}$  as  $k \rightarrow \infty$ ), so it suffices to prove that  $V^{\pi_{i+1,k+1}}(s) \geq V^{\pi_{i+1,k}}(s)$  for all  $k, s$ .

We do this by induction on  $k$ . The base case was handled above. By definition we have:

$$V^{\pi_{i+1,k+1}}(s) = R(s, \pi_{i+1}(s)) + \gamma \sum_{s'} \Pr_{s, \pi_{i+1}(s)}(s') V^{\pi_{i+1,k}}(s')$$

and

$$V^{\pi_{i+1,k}}(s) = R(s, \pi_{i+1}(s)) + \gamma \sum_{s'} \Pr_{s, \pi_{i+1}(s)}(s') V^{\pi_{i+1,k-1}}(s').$$

By induction, the RHS of the first equation is at least as large as the RHS of the second equation, so the LHS of the first equation must be at least as large as well.

4. **Sample complexity bounds.** For some learning algorithms, the hypothesis produced can be uniquely described by a small subset of  $k$  of the training examples. E.g., if you are learning an interval on the line using the simple algorithm “take the smallest interval that encloses all the positive examples,” then the hypothesis can be reconstructed from just the outermost positive examples, so  $k = 2$ . For a conservative Mistake-Bound learning algorithm, you can reconstruct the hypothesis by just looking at the examples on which a mistake was made, so  $k \leq M$ , where  $M$  is the algorithm’s mistake-bound. (In this case, you may also care about the *order* in which those examples arrived.)

Prove a PAC guarantee based on  $k$ . Specifically, fixing a description language (reconstruction procedure), so for a given set  $S'$  of examples we have a well-defined hypothesis  $h_{S'}$ , show that

$$\Pr_{S \sim D^n} \left( \exists S' \subseteq S, |S'| = k, \text{ such that } h_{S'} \text{ has 0 error on } S - S' \text{ but true error } > \epsilon \right) \leq \delta,$$

so long as

$$n \geq \frac{1}{\epsilon} \left( k \ln n + \epsilon k + \ln \frac{1}{\delta} \right).$$

Hint: Think of  $S'$  as a subset of indices, and imagine drawing points in  $S$  by drawing those in  $S'$  first.

Solution: Fix a subset  $S' \subseteq \{1, \dots, n\}$  of  $k$  indices and imagine drawing the examples in  $S'$  first. At this point,  $h_{S'}$  is now fully defined. If the true error of  $h_{S'}$  is greater than  $\epsilon$ , then the probability that  $h_{S'}$  makes zero mistakes on the remaining examples is at most  $(1 - \epsilon)^{n-k}$ . Formally, if  $A$  is the event in parentheses whose probability we wish to bound, and  $A_{S'}$  is the event “ $h_{S'}$  has error 0 on  $S - \{x_i : i \in S'\}$  but true error  $> \epsilon$ ” (so  $A = \bigcup_{S'} A_{S'}$ ), then we have  $\Pr(A_{S'}) \leq (1 - \epsilon)^{n-k}$ . Therefore,  $\Pr(A) \leq n^k (1 - \epsilon)^{n-k}$ , which is at most  $\delta$  for  $n$  satisfying the above inequality. Note that some algorithms (such as mistake-bound learning algorithms) may actually have their hypothesis depend on the *order* of points in  $S'$ , but that is still covered by the above argument.

Note the similarity of the form of this bound to VC-dimension and other bounds we have seen. These are often called “compression bounds”.