CSCI699: Theory of Machine Learning Fall 2023

Problem Set 3 (Theory portion)

Due: November 29 by 11:59 pm

Discussion is allowed and encouraged but everyone should write solutions on their own. Please also mention any collaborators you had substantial discussions with. You are also allowed to consult general resources on the internet (such as one of the books, or other lecture notes online), but you should not search for any the solutions themselves online.

If you use the LaTeX template, then please only keep your answers and remove the questions before submitting. Homeworks should be written in Latex and submitted via Gradescope. When you submit on Gradescope, make sure to mark the page which contains each answer.

Problem 1: Learning sparse parities

Consider the concept class of *sparse* parity functions:

$$\mathcal{C} = \{w(x) = \langle w, x\rangle \mod 2 : w \in \{0,1\}^d, \sum_{i=1}^{d} w_i = k\}.$$

Let the distribution D over x be the uniform distribution over $\{0,1\}^d$ for the remainder of this question.

(a) (5pts) Show that in the presence of random classification noise with noise level η, $\mathcal{C}$ is learnable (not necessarily efficiently) under the distribution D with $O\left(\frac{k \log d}{(1-2\eta)^2}\right)$ samples with high probability.

(b) (4pts) Show that then any SQ algorithm for learning $\mathcal{C}$ over the distribution D which makes queries of tolerance $\tau \geq \tau_{\min}$ must make $\Omega(\tau_{\min}^2 (d/k)^k)$ queries to $\mathrm{STAT}(c, D)$ to learn $\mathcal{C}$. The following theorem from the lecture notes will be useful.

Theorem 1. *If the concept class $\mathcal{C}$ has $\text{SQ-DIM}_D(\mathcal{C}) = s$, then any SQ algorithm for learning $\mathcal{C}$ over the distribution D which makes queries of tolerance $\tau \geq \tau_{\min}$ must make $\Omega(\tau_{\min}^2 s)$ queries to $\mathrm{STAT}(c, D)$ to learn $\mathcal{C}$.*

(c) (2pts) Contrast the bounds from the previous two parts. What can you say about the learnability of the sparse parity concept class?

Problem 2: Convergence rate of gradient descent for strongly convex problems

In this question, we will prove the exponential convergence rate of gradient descent for smooth, strongly convex functions which was stated in the lecture. Let f be a convex, differentiable function from $\mathbb{R}^d \to \mathbb{R}$. Further, assume that f is β-smooth and λ-strong convex:

Definition 2. *f is β-smooth if $\forall\, x, y \in domain(f)$,*

$$f(y) \leq f(x) + \langle y - x, \nabla f(x)\rangle + \frac{\beta}{2} \|y - x\|_2^2 .$$

Definition 3. *$f$is λ-strongly convex if $\forall\, x, y \in domain(f)$,*

$$f(y) \geq f(x) + \langle y - x, \nabla f(x)\rangle + \frac{\lambda}{2} \|y - x\|_2^2 .$$

(a) (2pts) Show that for any $w \in \mathbb{R}^d$,

$$\frac{\lambda}{2} \|w - w^*\|_2^2 \leq f(w) - f(w^*) \leq \frac{\beta}{2} \|w - w^*\|_2^2 .$$

Therefore we can relate the sub-optimality of w to its distance from w^*.

(b) (2pts) Let $w^* = \arg\min_{x \in \mathbb{R}^d} f(x)$. Show that for any $w \in \mathbb{R}^d$,

$$\frac{1}{2\beta} \|\nabla f(w)\|_2^2 \leq f(w) - f(w^*).$$

Hint: Use the definition of β-smoothness for $x = w$ and $y = w - \frac{1}{\beta}\nabla f(w)$.

This says that the norm of the gradient at a point is proportional to the sub-optimality of the point, and hence if gradient descent takes small steps then the current function value is close to optimal.

(c) (4pts) Suppose we run gradient descent with step size $1/\beta$, and let w_t be the gradient descent iterate at time t. Show that,

$$\|w_{t+1} - w^*\|_2^2 \leq \left(1 - \frac{\lambda}{\beta}\right) \|w_t - w^*\|_2^2 .$$

Hint: First use the gradient descent update step to write w_{t+1} in terms of w_t. Then expand the square, use the definition of strong-convexity, and the result from part (b) to simplify the expression.

(d) (2pts) Finally, show that for $\kappa = \beta/\lambda$,

$$f(w_t) - f(w^*) \leq e^{-t/\kappa} \|w_0 - w^*\|_2^2 (\beta/2).$$

Problem 3: Online learning of decision lists

In this question we consider the hypothesis class $\mathcal{H}$ of *decision lists*. A decision list is a function from $\{0,1\}^d \to \{0,1\}$, defined as follows. A decision list of length k over d Boolean variables $x_1, \ldots, x_d$ is a list of k pairs $\{(l_i, b_i), i \in [k]\}$ of literals l_i and bits b_i, and a single bit b_{k+1} (recall that a literal is either a Boolean variable x_i, or its negation $\bar{x}_i$). The output of a decision list is given by a if-then-else statement over the literals:

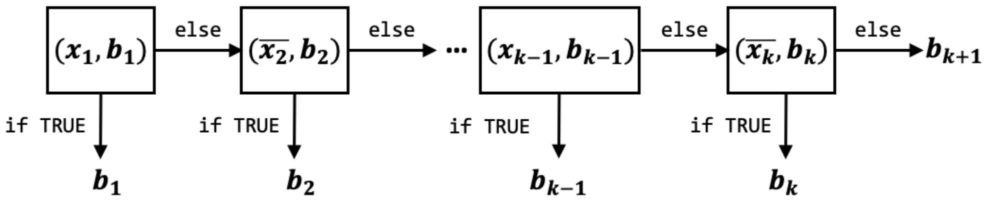

Figure 1: A decision list of length k.

To compute the value of $h(x)$ for any decision list h, we start from the first level of the list which has a literal l_1 and a bit b_1. If the literal l_1 evaluates to true, we output b_1, otherwise we go to the next level. Therefore the output of the decision list is b_i if the literal at the i-th level is the first literal which is satisfied, and is b_{k+1} if none of the literals are satisfied.

(a) (3pts) Show that the Littlestone dimension of the class of decision lists of level k on d variables is upper bounded by $O(k \log d)$. (Note that this only depends logarithmically on the number of variables d, and can hence handle a very large input space. An algorithm for learning decision lists which has a $\text{poly}(k, \log d)$ mistake bound is known as an *attribute-efficient* learning algorithm, since it is very efficient in the number of attributes).

(b) We will now show that there is an efficient algorithm A which learns $\mathcal{H}$ in the mistake bound model with a mistake bound $\mathcal{M}_A(\mathcal{H}) = O(dk)$.

(3pts) Using the following sketch, write an algorithm for learning decision lists:

- You can begin by putting all possible pairs $\{(l, b) : l \in \{x_i, \bar{x}_i, i \in [d]\}, b \in \{0,1\}\}$ at the first level of the decision list.
- At any time t, given any example a_t, start from the first level. If there is any pair (l, b) such that l is satisfied on the example, choose b as the output. If there is no such pair move to the next level.
- If the prediction is incorrect, you should move the chosen pair to some other level.

(Your algorithm need not be a proper learning algorithm, since multiple pairs could survive at every level.)

(c) (4pts) Show that the pair at level i for the ground truth decision list h^* is never demoted below the i-th level in your algorithm.

(d) (4pts) Finally, argue that your algorithm can only make $O(dk)$ mistakes.

Note that we get a $O(dk)$ mistake bound, whereas the Littlestone dimension is $O(k \log d)$. Learning decision lists with a $\text{poly}(k, \log d)$ mistake bound (attribute-efficient learning of decision lists) is a long-standing open problem in computational learning theory. We can also *ask this question in the PAC learning setup, where* $O(k \log d)$ *samples are information theoretically sufficient for learning, but there is no efficient algorithm with a* $\mathrm{poly}(k, \log d)$ *sample complexity. Check out [1, 2] to learn more about this problem if you are interested.*

References

[1] Adam R Klivans, Rocco A Servedio, and Dana Ron. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7(4), 2006.

[2] Philip Long and Rocco Servedio. Attribute-efficient learning of decision lists and linear threshold functions under unconcentrated distributions. *Advances in Neural Information Processing Systems*, 19, 2006.