CSC 665 Spring 2020 HW1[*]

Due by 11:59pm 3/18; submit your PDF file to gradescope.

Problem 1. Let $V = \{x \in \mathbb{R}^d : \|x\| \le U\}$. Define $\ell(x) = \ln(1 + \exp(-y\langle z, x\rangle))$ with $\|z\| \le 1$ and $y \in \{-1, 1\}$. Show that ℓ is $\exp(-U)$/-exp-concave. (tip: try the 1d case first, then try the generic d-dimensional case.)

[*] v6

Problem 2. Calculate the subdifferential set of the ϵ-insensitive loss: $f(x) = \max(|x - y| - \epsilon, 0)$. It is a loss used in regression problems where we do not want to penalize predictions x within $\pm\epsilon$ of the correct value y.

Problem 3. Let $\ell_t(x) = \frac{1}{2}(\langle z_t, x\rangle - y_t)^2$. Assume:

- $\|z_t\|_2 \le 1, \forall t$.
- $|y_t| \le Y, \forall t$ for some $Y > 0$.
- $V = \{v : \|v\|_2 \le Y\}$.

(a) In the class, we have discussed the ONS with the ℓ_t and V defined above and its regret bound, which was

$$\mathrm{Regret}_T(u) \le \frac{\lambda}{2}\|u\|_2^2 + 4Y^2 d \ln\left(1 + \frac{T}{8d\lambda}\right)$$

Recall that this was obtained after figuring out the α-exp-concaveness of ℓ_t, then figure out the μ-rank-one strong convexity (i.e., the equation (2) of the lecture note FTRL III). Please derive it again here, just as done in the lecture, starting from the ONS regret bound from the lecture note FTRL III (the one above Remark 3).

(b) Show that for all $x, y \in \mathbb{R}^d$,

$$\ell_t(u) = \ell_t(x) + \langle \nabla \ell_t(x), u - x \rangle + \frac{1}{2}(\langle z_t, u - x \rangle)^2$$

(c) Recall that we had a generalized notion of strong convexity involving a matrix A_t. What would be the proper A_t given our observation in 4.(b)?

(d) Based on the observations above, consider the following algorithm

$$x_t = \arg\min_{x \in V} \quad \frac{\lambda}{2}\|x\|_2^2 + \sum_{s=1}^{t-1} \ell_s(x) \ ,$$

which we call online ridge regression (ORR).

- Discuss how the objective function for computing x_t is different from the ONS in (a).
- Derive the final regret bound involving u, T, Y, d, λ only (hint: we already know a regret inequality involving A_t from the lecture, which is analyzed via FTRL with proximal regularizers).
- Is this regret bound different from (a) or not?

(e) Let us set $V = \mathbb{R}^d$ in (d). Let us call this version the unconstrained ORR (UORR).

- Can we arrive at the same regret bound as (a)? If not, what is the barrier? You may realize that you can still have 'some' regret bound, although it gives you less intuition.
- Open discussion: Under what situations would you expect that UORR enjoys more or less the same order of regret bound as (a)?
- Let us wear a practitioner's hat and say we do not care much about regret bounds. One obvious benefit is that you do not need to solve the constrained optimization and in fact we have a closed form solution. Besides that, can you find a scenario where UORR is better than (a)?

Now, programming assignments. Use your favorite programming language (e.g., python with numpy/scipy, matlab, julia, etc.). The final submission must be a single PDF file. FYI, there is a scanner anyone can use (it looks like a small fax machine) on Gould-Simpson the 7th floor copier room if needed.

Problem 4. (a) Implement:

- Online subgradient descent with the online-to-batch conversion (call it OSD-O2B), the particular scheme that gave us an “anytime” regret bound (Example 1 in the O2B lecture note).
- FTRL (Corollary 5 of the lecture note FTRL I, time-varying regularizer) with $\Psi(x) = \frac{1}{2}\|x\|_2^2$.
- OSD with adaptive stepsizes (AdaOSD; section 1 of adaptive algorithms lecture note), but let us wear a practitioner’s hat and say we do not perform projections, though it breaks the regret guarantee.
- UORR (from problem 3).

Remark: Sometimes you write the algorithm code, move one to evaluation, but the results are not convincing, and you don’t know where you have gone wrong; is it the data? bug in the algorithm code? or maybe the evaluation loop that weaves data and algorithms together? Mathematical codes are extremely hard to debug, so be sure to convince yourself that the algorithm code is correct (so when things are weird you don’t have to come back to the algorithm part). Repeatedly reading the code carefully also works, but you could also come up with a very simple dataset (1d/2d) and print out important variables. Monitor their convergence. You can also use a debugger that allows a step-by-step inspection of the variables (e.g., pdb/ipdb in python).

(b) Design your own data distribution $\mathcal{D}$ over (z, y) for the following two cases, respectively:

- Squared loss: $f(x, (z, y)) = (\langle z, x\rangle - y)^2$ with $y \in \mathbb{R}$.
- Hinge loss: $f(x, (z, y)) = \max(0, 1 - y\langle z, x\rangle)$ where $y \in \{-1, 1\}$.

Make suitable assumptions on the data (z, y) and generate the data using some random numbers (e.g., uniform, gaussian, etc.). Please describe your $\mathcal{D}$ for both cases.

(c) Let us take the stochastic optimization point of view where the goal is to solve the following approximately:

$$\min_x \mathbb{E}_{z,y\sim\mathcal{D}}[f(x, (z, y))] \ .$$

However, we only have access to stochastic versions $\ell_t(x) = f(x, (z_t, y_t))$ with $(z_t, y_t) \sim \mathcal{D}$. We can draw T data points and feed those to the algorithms, then monitor its convergence over $t \in \{1, \ldots, T\}$ on a *separate* dataset with M points drawn from $\mathcal{D}$ (pick M to be large so we get a stable estimate on $\mathbb{E}[\ell(x)]$). Choose T and M that makes sense to you.

For the squared loss:

- Test all 4 algorithms. Plot the hold-out average loss vs t of each algorithm in one plot. For each time t, you should use the suitable weighted average iterates recommended by the O2B technique (I believe you know which choice of weights gives you a reasonable regret bound).
- Plot the same as above but with the *last iterates* (i.e., for each time step t, the x_t output by the algorithm, not the ones computed by the O2B).
- Discuss the pros/cons of each method. Would you prefer using the last iterate? Justify it.

(d) Do the same as (c) for the hinge loss, but without UORR.