Convex Optimization: Homework 3

Instructor: Yuanzhi Li
Carnegie Mellon University
Due: April 19th 2020

1 Convexity of norm (10 points) [Stefani]

A norm $\|\cdot\|$ is a mapping that satisfies:

1. positive definiteness: $\forall\, x,\ \|x\| \geq 0$,
2. absolute homogeneity: $\forall\, \alpha \geq 0,\ \alpha\|x\| = \|\alpha x\|$,
3. triangle inequality: $\forall\, x, y,\ \|x + y\| \leq \|x\| + \|y\|$.

Show that $f(x) = \|x\|$ (any arbitrary norm) is a convex function.

2 Random Query for Convex Optimization (20 points) [Cinnie]

This problem studies the efficiency of the following optimization algorithm for a d-dimensional convex function f:

1. Fix a probability distribution $\mathcal{D}$, which does not depend on f. You don't know f, but let's say you somehow know that the optimum x^* has nonzero probability density over $\mathcal{D}$.
2. Sample $x_1, x_2, ...x_N \sim \mathcal{D}$
3. Return $x_m = \arg\min_{x_1,...x_N} f(x)$

Let $\mathcal{D}$ be the uniform distribution over the cube $\mathcal{Q} = \{x \mid -1 \leq x_i \leq 1\} = \{x \mid \|x\|_\infty \leq 1\}$.

Show that for every fixed $x_0 \in \mathbb{R}^d$, if we sample $N = d^{O(1)}$ points $x_1, x_2, \cdots, x_N$ uniformly from $\mathcal{Q}$, then with probability lower bounded by $1 - e^{-\Omega(d)}$, the following holds for all $i \in [N]$:

$$\|x_i - x_0\|_\infty > 0.99$$

Does this suggest that the above algorithm can be used to optimize a convex function in $\mathbb{R}^d$ in polynomial time with high probability, or not?

Hint for the probability bound: Let $\mathcal{S} = \{x : \|x - x_0\|_\infty \leq 0.99\}$. Find the probability that any given sample is in $\mathcal{S}$. Use this to union bound the probability that any of the N samples is in $\mathcal{S}$.

Hint for the implication on convex optimization: Consider what happens when the algorithm is used on $\|x - x_0\|_\infty$; the same principle applies for $\|x - x_0\|_2^2$ or any strictly convex function, though the proof is more complex without being more insightful.

3 Momentum (60 points + 40 bonus)

In this problem we attempt to arrive at a rudimentary understanding of the behavior of momentum based optimization methods.

Consider the function $f(x) = x^2$. Consider the following stochastic gradient model:

$$\widetilde{\nabla} f(x) = \nabla f(x)(1 + \xi)$$

Where ξ is an independent random variable such that $\xi \sim \mathcal{N}(0, \sigma^2)$ (and for different points x, x', or for the same point x queried at different iterations, the ξ's are also independent).

3.1 Using SGD (10 points) [Jerry]

Show that using learning rate $\eta \geq \frac{2}{\sigma}$ and $f(x_0) \neq 0$, the stochastic gradient descent update $x_{t+1} = x_t - \eta \widetilde{\nabla} f(x_t)$ satisfies that for as $t \to +\infty$,

$$\mathbb{E}[f(x_t)] \to \infty$$

In other words, the stochastic gradient descent update will diverge in expectation when using a large learning rate.

***Warning**: Each x_t is a random variable, so you should be using the law of total expectation if you want to relate x_{t+1} to x_t.*

3.2 GD with Momentum

It's quite difficult to analyze the theoretical behavior of momentum based algorithms on general families of stochastic convex optimization problems. If you are interested in these types of results, look up papers on Nesterov acceleration. In any case, the proof techniques they use are outside the scope of the course.

But it is possible to understand momentum algorithms in more restricted settings, which can serve as heuristic understanding. Let's analyze momentum updates on $f(x) = ax^2,\ a > 0$ when there is no stochasticity, i.e. $\sigma = 0$, equivalently $\widetilde{\nabla} f(x) = \nabla f(x)$.

3.2.1 Discrete updates (20 points) [Jerry]

Show that when $a = 1, \eta = 2, \gamma = 0.1$, gradient descent with momentum update

$$x_{t+1} = x_t - \eta g_t$$

$$g_t = (1 - \gamma) g_{t-1} + \gamma \nabla f(x_t),\ g_{-1} = 0$$

satisfies that for every t,

$$f(x_t) = f(x_0) e^{-\Omega(t)}$$

Hint: Consider a recurrence relation of the form $s_{t+1} = M s_t$ where $s_t = \begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix}$ for some $M \in \mathbb{R}^{2\times 2}$.

3.2.2 Continuous Updates Part 1 (15 points) [Cinnie]

We can also study what happens when we make the momentum parameter very small. This is useful when the problem is ill-conditioned, i.e. it has a high Lipschitz constant.

In particular, let $f(x) = ax^2$ and use gradient descent with momentum update $x_{t+1} = x_t - \eta g_t$ with g_t as defined in the previous sections. Additionally, define

- $\eta = \frac{\eta_0}{m}$
- $\gamma = \frac{\gamma_0}{m}$
- $\tau(t) = \frac{t}{m}$
- $d\tau(t) = \tau(t+1) - \tau(t)$
- $h(t) = x_t$
- $dh(t) = h(t+1) - h(t)$
- $d^2h(t) = dh(t+1) - dh(t)$

Do not assume $\eta = 2$ or $\eta_0 = 2$, keep these as variables.

Find coefficients c_1 and c_2, such that

$$\frac{d^2h(t)}{d\tau(t)^2} = c_1(a, \eta_0, \gamma_0)\frac{dh(t)}{d\tau(t)} + c_2(a, \eta_0, \gamma_0)h(t) + \epsilon$$

Here, ϵ may depend on $m, a, \eta_0, \gamma_0, \frac{dh(t)}{d\tau(t)}, h(t)$ and has the property:

$$\lim_{m\to\infty} \epsilon = 0$$

You may assume that $\frac{dh(t)}{d\tau(t)}$ and $h(t)$ converge to finite values for large m.

3.2.3 Continuous Updates Part 2 (15 points) [Jerry]

Treat this expression as a differential equation:

$$\frac{d^2h}{d\tau^2} = c_1(a, \eta_0, \gamma_0)\frac{dh}{d\tau} + c_2(a, \eta_0, \gamma_0)h$$

with initial values:

$$h(0) = x_0,\ \frac{dh}{d\tau}(0) = 0$$

For the following scenarios, use your favorite tool (e.g. WolframAlpha) to solve for the solution analytically (not numerically), and

- Type out the analytical solution
- Plot the continuous estimate $h(\tau)$ for $\tau \in [0, 200]$ (use $\eta_0 = \eta$, $\gamma_0 = \gamma$)
- Plot the true values x_t for $t = 0...200$

The scenarios are:

- $x_0 = 1, a = 1, \eta = \frac{1}{64}, \gamma = \frac{1}{4}$

- $x_0 = 1, a = 1, \eta = \frac{1}{64}, \gamma = \frac{1}{8}$
- $x_0 = 1, a = 1, \eta = \frac{1}{64}, \gamma = \frac{1}{16}$
- $x_0 = 1, a = 901, \eta = \frac{1}{64}, \gamma = \frac{1}{8}$

Compare the continuous estimates with the true values, under what kinds of settings does the estimate start to break down?

For a given a and η, how does the behavior of the algorithm differ for smaller γ vs. larger γ?

3.3 Tying it together (all bonus questions!) [Jerry]

3.3.1 Discrete case (20 bonus points)

Given $f(x) = ax^2$ with learning rate $\eta > 0$ and $x_0 \neq 0$, let $\mathcal{G}$ be the set of values for γ where gradient descent with momentum converges. Show that $\sup \mathcal{G} = \frac{2}{a\eta+1}$. In other words, show that all $\gamma > \frac{2}{a\eta+1}$ does not converge, and that you can find γ arbitrarily close to $\frac{2}{a\eta+1}$ which converges.

Hint: You can prove this by building the same M matrix as in Q3.2.1, but using more indirect reasoning to show the two parts of the proof. It's not hard to fully analyze the system in Q3.2.1, but doing that here is a huge pain. It may help to know that $\gamma = \frac{2}{a\eta+1}$ eventually results in the iterates alternating between two values of opposite sign.

3.3.2 Condition number (20 bonus points)

Consider the function $f(x, y) = ax^2 + y^2$, where $a > 1$.

Note that performing gradient descent with momentum on this function is the same as running it independently for $f_1(x) = ax^2$ and $f_2(y) = y^2$.

Assume that if $0 < \gamma < \frac{1}{a\eta+1}$ then the continuous limit is a good model of the algorithm's behavior for both coordinates.

Show that η, γ can be chosen so that $(x_t, y_t) = O((1 - R)^t(x_0, y_0))$ where $R = \Omega\left(\frac{1}{\sqrt{a}}\right)$.

For comparison, normal gradient descent achieves $R = \Omega\left(\frac{1}{a}\right)$ for this function, which is much worse for large a. A similar story holds for any strongly convex function.

Hint: look up the general form for the solution of the differential equation in Q3.2.2

4 Adagrad: The simple case (10 points) [Stefani]

Show that for function $f(x) = f(x_1, x_2) = ax_1^2 + x_2^2$ for $x \in \mathbb{R}^2$ and $a > 0$, if we use Adagrad to optimize f with learning rate $\eta = 1$, derive the ratio (as a function of a) of the pre-conditioner:

$$\gamma = \sqrt{\frac{\sum_{s \leq t}[\nabla f(x_s)]_1^2}{\sum_{s \leq t}[\nabla f(x_s)]_2^2}},$$

given a fixed initial point $x_0 = (c, c)$.

5 Non-convex optimization (50 points, coding)

Considering maximizing the function $f(x) = x + \sin(6x)$ over interval $\mathcal{D} = [-5, 5]$.

5.1 Gradient Ascent (10 points) [Stefani]

- Starting from the point $x_0 = 0$, implement the gradient ascent algorithm with step sizes $\eta \in \{0.01, 0.02, 0.05\}$ for 100 steps.
- Plot $f^\star - f(x_t)$ versus the iteration number t where $f^\star$ is the maximum of f on $\mathcal{D}$ for all step sizes on the same plot.
- Plot $|x_t - x^\star|$ (in log-scale) versus the iteration number t where $x^\star$ is the maximizer of f on $\mathcal{D}$ for all step sizes on the same plot.

The goal is to visually compare and contrast the variation in convergence rates. Please write 1-2 sentences (that's all!) explaining your plots.

5.2 Bayesian Optimization (5 points + 20 points) [Vishwak]

5.2.1 Expected Improvement (5 points)

For $P_t(x) = \mathcal{N}(\mu_t(x), \sigma_t^2(x))$, show that the expected improvement function

$$\text{EI}(x) = \mathbb{E}_{g(x)\sim P_t}[[g(x) - f_t^*]^+]$$

has a closed form expression:

$$\text{EI}(x) = (\mu_t(x) - f_t^*)\Phi\left(\frac{\mu_t(x) - f_t^*}{\sigma_t(x)}\right) + \frac{\sigma_t(x)}{\sqrt{2\pi}}\exp\left(-\frac{(f_t^* - \mu_t(x))^2}{2\sigma_t^2(x)}\right)$$

5.2.2 Implementation (20 points)

The initial point is $x_0 = 0$.

- Use the Bayesian optimization algorithm taught in class, with a Gaussian Kernel (with variance σ^2), for $t = 100$ iterations.
- Try different $\sigma^2 = \{0.03, 0.1, 0.3, 1, 3, 10, 30\}$
- Plot the mean value function $\mu_t(x)$ and the confidence band $[\mu_t(x) - \sigma_t(x), \mu_t + \sigma_t(x)]$ for $t \in \{1, 21, 41, 61, 81, 99\}$.
- On the same plot, plot $x_{t'}$ for $t' \leq t$. For example, for $t = 61$, plot x_1, x_{21}, x_{41} and x_{61} alone.
- Report the final value obtained at iteration 100 i.e., $f_{100}^* = \max_{t\in\{1,2,\ldots,100\}} f(x_t)$

It is strongly recommended that the plots be separate i.e., for each σ^2 and each t, you have a separate plot.

Please write 1-2 sentences (that's all!) explaining your plots.

Hint: When optimizing the acquisition function, you can use either gradient ascent (computing the gradient by computing the difference of function value between two near by points for sufficient large number of iterations - say 1000), or using brute force search.

5.3 Simulated Annealing (20 points) [Vishwak]

Use the Simulated Annealing algorithm taught in class (use density $e^{\lambda f(x)}$ for maximization), with initial $\lambda_0 = 0.01$ and stop until $\lambda_t \geq 100$. Use $\eta = 0.1$ and $\sigma^2 = 0.1$. Pick a proper number of iterations T for the Metropolis-Hasting algorithm.

If a point is outside $\mathcal{D}$ after sampling from the Gaussian distribution, reject it immediately.

Suppose for each λ_t, the Metropolis-Hasting algorithm picks points $x_1^{(t)}, x_2^{(t)}, \cdots, x_T^{(t)}$. Plot the following values vs t:

$$\frac{1}{T}\sum_{s\in[T]} x_s^{(t)}, \quad \frac{1}{T}\sum_{s\in[T]} f(x_s^{(t)})$$

Please write 1-2 sentences (that's all!) explaining your plots.