Constrained Optimization and SGD

Optimization for Machine Learning — Homework #2

Monday 12th June, 2023

The theory part can be handed-in physically during the exercise session, or digitally on Moodle. The programming part has to be sent on Moodle. *Group work is allowed (2 – 3 people), but submissions are personal..*

Part I: Theory *12 points*

I.1. Constrained optimization

Exercise I.1 (Constrained optimization, 5 points)**.** You encounter the following optimization problem on $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$:

$$\begin{array}{rl} \text{minimize} & f(x) := -(x_1 + x_2) \\ \text{subject to} & c(x) := x_1^2 + x_2^2 \leqslant 1 \end{array}$$

1. What is the dimension of the Lagrangian multiplier α? Write down the Lagrangian $L(x, \alpha)$.
2. Show that the dual function $g(\alpha) := \min_x L(x, \alpha)$ is $g(\alpha) = -(\alpha + \frac{1}{2\alpha})$.
3. For feasible α and x (meaning $\alpha \geqslant 0$ and $c(x) \leqslant 0$), show that $g(\alpha) \leqslant f(x)$.
4. Solve the dual problem

$$\begin{array}{rl} \text{maximize} & g(\alpha) \\ \text{subject to} & \alpha \geqslant 0 \end{array}$$

5. What is the solution to the original problem?

I.2. General analysis

Exercise I.2 (Inequality of c-strongly convex functions, 2 points)**.** Recall that, given $c > 0$, a function $E\colon \Omega \subset \mathbb{R}^d \to \mathbb{R}$ is called c-strongly convex if Ω is convex and

$$\forall (x,y) \in \Omega^2, \quad E(y) \geqslant E(x) + \langle \nabla E(x), y - x \rangle + \frac{c}{2}\|y - x\|_2^2$$

A strongly convex function has a unique minimizer $x_* = \operatorname{argmin}_{x\in\Omega} E(x)$.
Show the that, if E is c-strongly convex, it satisfies the following inequality:

$$\forall y \in \Omega, \quad 2c(E(y) - E_*) \leqslant \left\|\nabla E(y)\right\|_2^2$$

Hint: Study the function $q(y) = E(x) + \langle \nabla E(x), y - x\rangle + \frac{c}{2}\|y - x\|_2^2$.

I.3. SGD Analysis

Exercise I.3 (5 points)**.** Recall that if we assume that F is strongly convex, we can show that the gradient descent converges with rate $\mathcal{O}(\rho^k)$ where $0 < \rho < 1$, and k is the number of iterations. This rate is called "linear convergence".

Assume we have a L-smooth and c-strongly convex function. Recall the expression for the convergence of stochastic gradient descent:

$$\begin{aligned}\mathbb{E}[\|\nabla f(x_T)\|^2] &\leq 2\left(\frac{2\sqrt{T+1}-1}{L}\right)^{-1} \cdot \left(\mathbb{E}[f(x_0)] - f^* + \frac{\log(T)+1}{L^2}\right) \\ &= \mathcal{O}\left(\frac{\log(T)}{L\sqrt{T}}\right),\end{aligned}$$

where $f^* = \min_x f(x)$, α is the step size, L is the Lipschitz constant, and T is the total number of iterations.

1. How does this compare to the expression that we get for the gradient descent?
2. Derive the rate of convergence for the stochastic gradient descent for strongly convex functions.

Hints:

1. Start from the expression, valid when f is L-smooth:
$$\mathbb{E}[f(x_{k+1})] \leq \mathbb{E}[f(x_k)] - \frac{\alpha}{2}\mathbb{E}[\|\nabla f(x_k)\|^2\|] + \frac{\alpha^2\sigma^2 L}{2}.$$
2. Apply the inequality for c-strongly convex functions (Polyak-Łojasewicz inequality):
$$\|\nabla f(x)\|^2 \geq 2c(f(x) - f^*), \qquad c > 0$$
3. Subtract from both sides f^*.
4. Subtract from both sides the fixed point $\frac{\alpha^2\sigma^2 L}{2c\alpha}$
5. Apply the previous step recursively for T steps.
6. Use the inequality $(1 - c\alpha) \leq \exp(-c\alpha)$.

Part II: Programming *8 points*

Exercise II.1 (SVM with SGD, 8 points)**.** In this exercise, (linear) SVM will be solved with SGD.

We consider the SVM problem with prediction $h(x,w) := \langle x, w\rangle + b$ and (differentiable) loss $\ell(\hat{y}, y) := \frac{1}{10}\ln(1+\exp(10(1-y\cdot\hat{y})))$. The composed loss is denoted by $f(w,(x,y)) := \ell(h(x,w),y)$. Given a training dataset $\{(x_i,y_i)\}_{i\in[n]}$, the empirical risk is $R_n(w) := \frac{1}{n}\sum_{i=1}^{n} f(w,(x_i,y_i))$.

The stochastic gradient descent (SGD) algorithm is given in Algorithm 1. The random variable ξ_k selects samples and their targets (i.e. elements from $\mathcal{X}\times\{-1,1\}$).

Algorithm 1 Stochastic Gradient Descent [1, Algorithm 4.1].

Choose an initial iterate w_1
for $k = 1, 2, \dots$ **do**
  Generate a realization of the random variable ξ_k with values in $\mathcal{X}\times\{-1,1\}$ (e.g. batch of samples)
  Compute a stochastic vector $g(w_k,\xi_k)$
  Choose a step size $\alpha_k > 0$
  Set the new iterate as $w_{k+1} \leftarrow w_k - \alpha_k g(w_k,\xi_k)$.
end for

1. What is the role of the factor 10 in ℓ?
2. Is the function $w \mapsto R_n(w)$ (strongly) convex? L-smooth? What is the gradient $\nabla_w \ell(h(x,w),y)$?
3. Implement the SGD algorithm Algorithm 1, with data given by `utils.gen_linsep_data`. You can choose how to sample from the dataset, either one sample at a time or using a batch of samples.
4. Test different step sizes and show the convergence.

Answer II.1.

1. The factor $p = 10$ in ℓ controls how "close" the function ℓ is to $f\colon z \mapsto \max(0, 1-z)$: with higher p, the function is closer to f. See it here.
2. We see that R_n is convex as addition of convex functions.

   To see if the function R_n is strongly convex or L-smooth, one can first study the Hessian of $f\colon z \mapsto \frac{1}{10}\ln(1+\exp(10(1-z)))$. One computes $f''(z) = \frac{10\exp(10(1-z))}{(1+\exp(10(1-x)))^2}$.

   The function f is not c-strongly convex: since $\lim_{z\to-\infty} f(z) = \lim_{z\to+\infty} f(z) = 0$, there is no $c > 0$ such that $\forall z \in \mathbb{R},\ f''(z) \geqslant c$.

   The function f is L-smooth: since $\lim_{z\to-\infty} f(z) = \lim_{z\to+\infty} f(z) = 0$ and f'' is continuous, there exists $L \in \mathbb{R}$ such that $\forall z \in \mathbb{R},\ f''(z) \leqslant L$.

   Since R_n is the composition of affine functions of w with f, we can draw the conclusion on R_n as well. It is not c-strongly convex, and it is L-smooth. (Warning: the constant L is not necessarily the same one as the one for f).

To compute it more precisely, one can compute the Hessian of R_n and find the bounds (c, L) such that $cI \preccurlyeq \nabla^2 R_n(w) \preccurlyeq LI$.

The gradient of the loss ℓ with respect to w is (chain rule):

$$\nabla_w \ell(h(x;w), y) = \frac{\partial h(x;w)}{\partial w}^\top \ell'(h(x;w), y) = -x \cdot y \cdot \frac{\exp(10(1 - y \cdot h(x;w)))}{1 + \exp(10(1 - y \cdot h(x;w)))}.$$

References

[1] Léon Bottou, Frank E. Curtis and Jorge Nocedal. "Optimization Methods for Large-Scale Machine Learning". 2016. DOI: 10.1137/16M1080173. arXiv: 1606.04838.