EE-556: Mathematics of Data **Fall 2019**

Homework #1 - Due date: 1st November 2019

Student: Oriol Barbany Mayor

Problem 1 - Geometric properties of the objective function f

(a) By linearity of the gradient operator, we have that

$$f(\mathbf{x},\mu) := \frac{1}{n}\sum_{i=1}^{n} g_i(\mathbf{x},\mu) + \frac{\lambda}{2}\|\mathbf{x}\|^2 \Longrightarrow \nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\nabla g_i(\mathbf{x},\mu) + \lambda\mathbf{x} \tag{1}$$

$$\nabla g_i(\mathbf{x},\mu) = \begin{cases} -b_i\mathbf{a}_i, & b_i\mathbf{a}_i^T\mathbf{x} \leq 0 \\ b_i\mathbf{a}_i(b_i\mathbf{a}_i^T\mathbf{x}-1), & 0 < b_i\mathbf{a}_i^T\mathbf{x} \leq 1 \\ 0, & 1 \leq b_i\mathbf{a}_i^T\mathbf{x} \end{cases} \tag{2}$$

which yields

$$\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\left[-b_i\mathbf{a}_i\mathbb{1}_{\{b_i\mathbf{a}_i^T\mathbf{x}\leq 0\}} + b_i\mathbf{a}_i(b_i\mathbf{a}_i^T\mathbf{x}-1)\mathbb{1}_{\{0<b_i\mathbf{a}_i^T\mathbf{x}\leq 1\}}\right] + \lambda\mathbf{x} \tag{3}$$

Note that with the proposed notation, $\mathbf{I}_L$ selects the coordinates that are in the linear region (first case of (2)), and $\mathbf{I}_Q$ the ones that lie in the quadratic region (second case). If we express (3) in matrix form using $\tilde{\mathbf{A}}$ as given in the problem description and the previous matrices, we get

$$\nabla f(\mathbf{x}) = -\frac{1}{n}\tilde{\mathbf{A}}\mathbf{I}_L\mathbf{1} + \frac{1}{n}\tilde{\mathbf{A}}^T\mathbf{I}_Q[\tilde{\mathbf{A}}\mathbf{x}-\mathbf{1}] + \lambda\mathbf{x} \tag{4}$$

Let's now compute the Lipschitz constant of the gradient:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| = \left\|\lambda(\mathbf{x}-\mathbf{y}) + \frac{1}{n}\tilde{\mathbf{A}}^T\mathbf{I}_Q\tilde{\mathbf{A}}(\mathbf{x}-\mathbf{y}) - \frac{1}{n}\tilde{\mathbf{A}}\mathbf{I}_L\mathbf{1} + \frac{1}{n}\tilde{\mathbf{A}}\mathbf{I}_L\mathbf{1}\right\| \tag{5}$$

$$= \left\|\left(\lambda\mathbf{I} + \frac{1}{n}\tilde{\mathbf{A}}^T\mathbf{I}_Q\tilde{\mathbf{A}}\right)(\mathbf{x}-\mathbf{y})\right\| \tag{6}$$

$$\leq \left\|\lambda\mathbf{I} + \frac{1}{n}\tilde{\mathbf{A}}^T\mathbf{I}_Q\tilde{\mathbf{A}}\right\|\|\mathbf{x}-\mathbf{y}\| \tag{7}$$

where the inequality follows from definition of the spectral norm.

Using the fact that $\lambda\mathbf{I} + \frac{1}{n}\tilde{\mathbf{A}}^T\mathbf{I}_Q\tilde{\mathbf{A}} = \lambda\mathbf{I} + \frac{1}{n}\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}$ given in the problem statement and using the triangle inequality, we get that

$$\left\|\lambda\mathbf{I} + \frac{1}{n}\tilde{\mathbf{A}}^T\mathbf{I}_Q\tilde{\mathbf{A}}\right\| = \left\|\lambda\mathbf{I} + \frac{1}{n}\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\right\| := \max_{\mathbf{x}:\|\mathbf{x}\|=1}\left\|\left(\lambda\mathbf{I} + \frac{1}{n}\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\right)\mathbf{x}\right\| \tag{8}$$

$$\leq |\lambda|\max_{\mathbf{x}:\|\mathbf{x}\|=1}\|\mathbf{x}\| + \frac{1}{n}\max_{\mathbf{x}:\|\mathbf{x}\|=1}\left\|\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\mathbf{x}\right\| := \lambda + \frac{1}{n}\left\|\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\right\| \tag{9}$$

$$\leq \lambda + \frac{1}{n}\left\|\tilde{\mathbf{A}}^T\right\|\left\|\tilde{\mathbf{A}}\right\| \tag{10}$$

where the last inequality again follows from definition of the spectral norm and I assume that λ is non-negative. The proof is completed using the fact that $\left\|\tilde{\mathbf{A}}\right\| = \|\mathbf{A}\|$ and $\left\|\tilde{\mathbf{A}}^T\right\| = \|\mathbf{A}^T\|$.

(b) Assuming that all the samples lie in the quadratic region, i.e. $\mathbf{I}_Q = \mathbb{I}$ and $\mathbf{I}_L = \mathbf{0}$:

$$\nabla f(\mathbf{x}) = \frac{1}{n}\tilde{\mathbf{A}}^T[\tilde{\mathbf{A}}\mathbf{x} - \mathbf{1}] + \lambda\mathbf{x} \tag{11}$$

The gradient ∇f exists in all domain of $\mathbf{x}$ and correspond to a linear function on $\mathbf{x}$, which is infinitely differentiable. We then say that f is twice differentiable (in also fact infinitely differentiable and thus twice), and its hessian corresponds to

$$\nabla^2 f(\mathbf{x}) = \frac{1}{n}\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbb{I} = \frac{1}{n}\mathbf{A}^T\mathbf{A} + \lambda\mathbb{I} \tag{12}$$

where the equality follows since $b_i^2 = 1\ \forall i$, which implies that $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} = \mathbf{A}^T\mathbf{A}$.

(c)

Lemma 1. *A function f is $\mu-$strongly convex iff $\nabla^2 f(\mathbf{x}) \succeq \mu\mathbb{I}$*

Proof. As seen in lectures, f is $\mu-$strongly convex iff $g(\mathbf{x}) := f(\mathbf{x}) - \frac{\lambda}{2}\|\mathbf{x}\|$ is convex, and hence if $\nabla^2 g(\mathbf{x}) \succeq 0$. The proposition follows by combining these two. □

Given that we have the hessian of f, we can use Lemma 1 to compute the strong convexity constant:

$$\mathbf{x}^T\nabla^2 f(\mathbf{x})\mathbf{x} = \lambda + \frac{1}{n}\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} \geq \lambda := \mu \quad \forall\mathbf{x} \tag{13}$$

where the inequality follows since $\mathbf{A}^T\mathbf{A}$ is positive semidefinite. This latter is true since we can define $\mathbf{y} := \mathbf{A}\mathbf{x}$ and $\|y\| := \mathbf{y}^T\mathbf{y} := \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$ will be non-negative by definition of the norm.

Problem 2 - First order methods for SVM

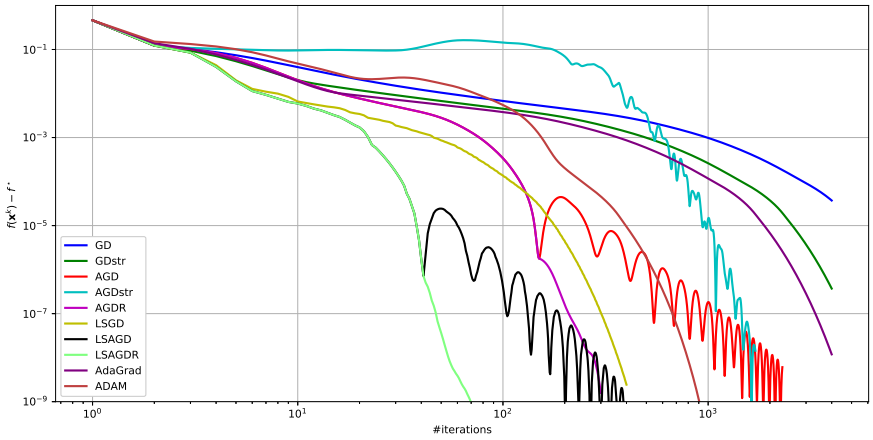

Figure 1: Comparison in number of iterations for deterministic first-order methods

(a) **GD and GDstr:** As already expected, the gradient descent considering both the strong convexity of the function to optimize and its smoothness converges faster than that only considering smoothness (see Figure 1). Moreover, the cost of each iteration is exactly the same since we are doing the same number of computations but only changing the learning rate. This can be seen in Figure 2.

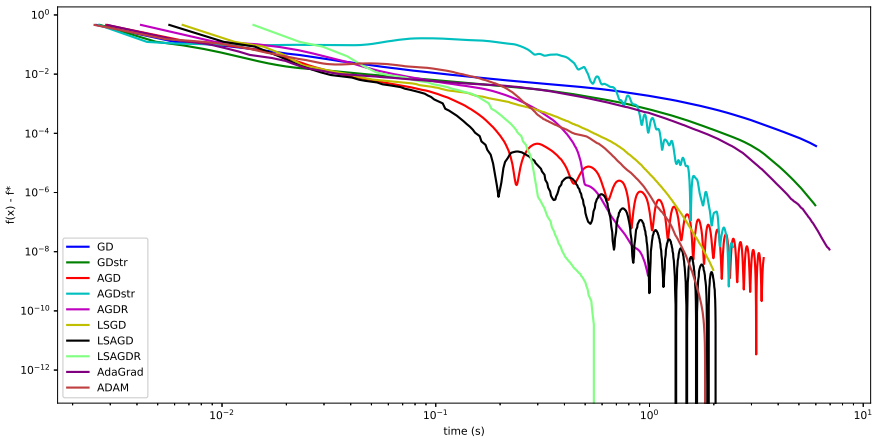

Figure 2: Comparison in time for deterministic first-order methods

(b) **AGD and AGDstr:** Unlike the previous case, the Accelerated Gradient Descent algorithm works worse when we use information about the strong convexity of the function. Moreover, the cost per iteration is essentially the same as in the previous case, so this algorithms beat naive gradient descent in both number of iterations and time to converge.

(c) **LSGD and LAGD:** We converge with less iterations when we use the previous procedures with a step size that takes into account the local geometry. This step size is found by a linear search procedure, which translates into more cost (and thus more time) per iteration. Nevertheless, in this case we can see that the overhead is not significant enough for the previous algorithms to beat this version at almost any time.

(d) **AGDR and LAGDR:** The drawback of accelerated methods are that they are not monotone. In fact we can see the so-called Nesterov Ripples (oscillations) caused by the high momentum term, which is restarted in the algorithms presented in this section every time we increase the cost after one accelerated gradient step.

For this reason, each of these algorithms is exactly the same as their non-restart version until the first oscillation as seen in Figure 1. By construction, the restart version is now monotonic and thus the objective value decreases much faster than with the non-restart version.

Nevertheless, note that iterations are slightly more costly with the restart version, specially when we actually restart since we compute two gradient steps in this iteration. Hence, in Figure 2, we don't see the non-restart and restart versions sticking together for the first iterations as in Figure 1.

(e) **AdaGrad:** This algorithm adapts to the local geometry of the problem with first-order information and doesn't need the Lipschitz constant as all the previous algorithms do nor the strong convexity constant as SGDstr and AGDstr. As seen in Figures 1 and 1, this algorithm converges faster than Gradient Descent both in the number of iterations and in time, even if Gradient Descent uses an step size that is optimized for the Lipschitz and strong convexity constants of the function to optimize. Moreover, this algorithm introduces two new hyper-parameters, whose tuning sometimes could make a big difference.

(f) **ADAM:** Even if this algorithms can fail to converge to the global minimum of a convex problem, it usually performs well. Indeed in this case it converges faster that the previous adaptive algorithm and even faster than AGD knowing the Lipschitz constant. ADAM introduces four new hyper-parameters and thus depending on the time that we have to tune them or the performance in some problem with

them taking the usual values, we may prefer other algorithms even if ADAM could achieve better results with the right tuning.

Problem 3 - Stochastic Gradient methods for SVM

(a) Since we choose each index i_k uniformly at random from all the n samples, we have that

$$\mathbb{E}[\nabla f_{i_k}(\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\mathbf{x}) \tag{14}$$

$$= \lambda\mathbf{x} + \frac{1}{n}\sum_{i=1}^{n}\left[-b_i\mathbf{a}_i\mathbb{1}_{\{b_i\mathbf{a}_i^T\mathbf{x}\leq 0\}} + \mathbf{a}_i(\mathbf{a}_i^T\mathbf{x} - b_i)\mathbb{1}_{\{0<b_i\mathbf{a}_i^T\mathbf{x}\leq 1\}}\right] \tag{15}$$

$$:= \nabla f(\mathbf{x}) \tag{16}$$

where the last equality is the one already found in (2).

Assumption 1. *Sample i_k lies in the quadratic region*

Taking Assumption 1, we have that

$$\|\nabla f_{i_k}(\mathbf{x}) - \nabla f_{i_k}(\mathbf{y})\| = \left\|\lambda(\mathbf{x}-\mathbf{y}) + \mathbf{a}_{i_k}\mathbf{a}_{i_k}^T(\mathbf{x}-\mathbf{y})\right\| \leq \left\|\lambda\mathbb{I} + \mathbf{a}_{i_k}\mathbf{a}_{i_k}^T\right\|\|\mathbf{x}+\mathbf{y}\| \tag{17}$$

where the inequality follows by definition of the spectral norm.

Using triangle inequality as in problem 1 gives

$$\left\|\lambda\mathbb{I} + \mathbf{a}_{i_k}\mathbf{a}_{i_k}^T\right\| \leq \lambda + \left\|\mathbf{a}_{i_k}\mathbf{a}_{i_k}^T\right\| := \lambda + \max_{\mathbf{x}:\|\mathbf{x}\|=1}\left\|\mathbf{a}_{i_k}\mathbf{a}_{i_k}^T x\right\| := \lambda + \max_{\mathbf{x}:\|\mathbf{x}\|=1}\sqrt{x^T\mathbf{a}_{i_k}\mathbf{a}_{i_k}^T\mathbf{a}_{i_k}\mathbf{a}_{i_k}^T x} \tag{18}$$

$$:= \lambda + \max_{\mathbf{x}:\|\mathbf{x}\|=1}\sqrt{x^T\left\|\mathbf{a}_{i_k}\right\|^2\left\|\mathbf{a}_{i_k}\right\|^2 x} = \lambda + \left\|\mathbf{a}_{i_k}\right\|^2\max_{\mathbf{x}:\|\mathbf{x}\|=1}\|x\| = \lambda + \left\|\mathbf{a}_{i_k}\right\|^2 \tag{19}$$

Note that assumption 1 is w.l.o.g. since if we lie on the linear region,

$$\|\nabla f_{i_k}(\mathbf{x}) - \nabla f_{i_k}(\mathbf{y})\| = \|\lambda(\mathbf{x}-\mathbf{y})\| = \lambda\|\mathbf{x}-\mathbf{y}\| < L\|\mathbf{x}-\mathbf{y}\| \tag{20}$$

hence global smoothness is the same (valid upper bound) even if it's not locally tight for the samples in the linear region.

(b) **SGD:** This algorithm has become the workhorse for training supervised machine learning problems which have a sum-structured objective function since the cost per iteration can be up to n times cheaper than that of GD, where n is the number of samples. When comparing to GD with respect to the number of iterations (Figure 3), we see that the SGD requires a lot more iterations to converge since at each iteration we are only considering one random sample instead of the whole data as GD does. Nevertheless, the turns change when we compare these algorithms in time. As seen in Figure 4, before GD performs the first iteration, SGD has achieved an objective value that can be good enough for some applications or even be used prior to methods that need a close start such as Newton's.

(c) **SAG:** When averaging stochastic gradients, we eventually get something similar to Gradient Descent if the variation of each gradient from one iteration to another is tiny (which happens near to a minimum). Thus, we can see that in comparison to SGD, we get better performance for the last iterations with exactly the same cost per iteration.

(d) **SVR:** This algorithm is the one that performs the best among all the ones seen before with respect to the number of iterations, since it essentially converges in only one iteration. Nevertheless, the variance reduction incurs an overhead at each iteration, making its cost $\mathcal{O}(1000L_{\max})$ times that of SGD. For this reason, we can see in Figure 4 that when SVR performs the first iteration, SGD has already done more than 2500 iterations.

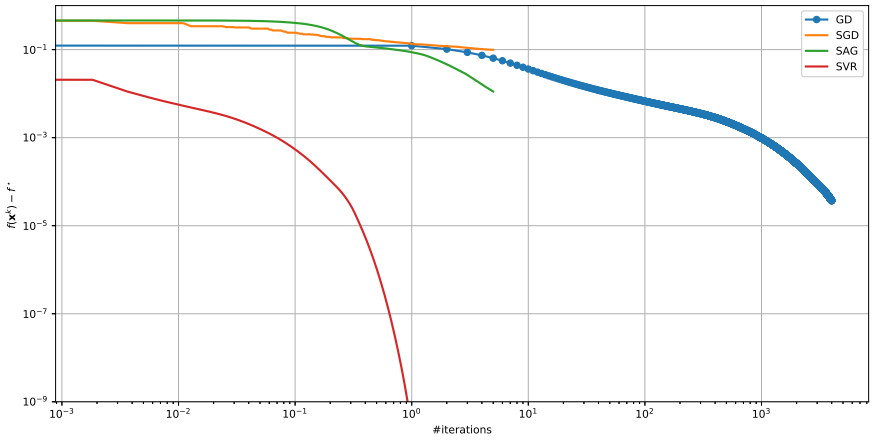

Figure 3: Comparison in number of iterations for stochastic first-order methods

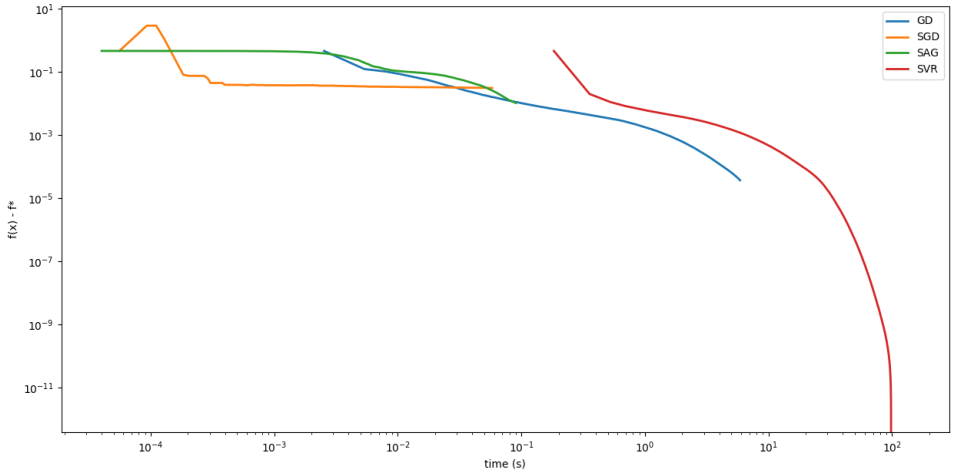

Figure 4: Comparison in time for stochastic first-order methods