Optimization for Data Science
June 26, 2018

1. (6 POINTS) Describe in depth the Gauss-Southwell Block-Coordinate Gradient Descent method.

   Solution 1. See Notes Section 4.5.1.

2. (7 POINTS) Describe in depth Away-Step, Pairwise and Fully Corrective Frank-Wolfe. Furthermore, prove that Fully Corrective Frank-Wolfe converges in a finite number of steps (when minimizing a convex function over a polytope).

   Solution 2. See Notes Section 5.3 and 5.4.

3. (7 POINTS) Consider the problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2,$$

with $A \in \mathbb{R}^{m \times n}$ and $m \leq n$. Calculate the gradient related to the objective function, then explain the differences, in terms of computational cost per iteration, between the classic gradient method and a BCGD method (with blocks of dimension 1) when solving the problem.

Solution 3. The gradient in this case is

$$\nabla f(x) = 2A^\top (Ax - b).$$

At each iteration of the gradient method we have to calculate

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k).$$

Hence we have

$$\nabla f(x_{k+1}) = 2A^\top (Ax_{k+1} - b).$$

The total cost is given by the matrix-vector product Ax_{k+1} and the product $A^\top r(x_{k+1})$, where $r(x_{k+1}) = Ax_{k+1} - b$. Hence, we need two matrix-vector products to get the new gradient value. If we now consider BCGD, we can write

$$x_{k+1} = x_k - \alpha_k \nabla_{i_k} f(x_k) e_{i_k}.$$

Now, if we set $g_{i_k} = \alpha_k \nabla_{i_k} f(x_k)$, we get

$$\begin{aligned} \nabla f(x_{k+1}) &= 2A^\top (Ax_{k+1} - b) = 2A^\top \left[A\left(x_k - g_{i_k} e_{i_k}\right) - b\right] = \\ &= \nabla f(x_k) - 2g_{i_k} A^\top (Ae_{i_k}) = \nabla f(x_k) - 2g_{i_k} A^\top A_{i_k}. \end{aligned}$$

This means that to calculate the full gradient (when considering a BCGD-like step) we perform only one matrix-vector multiplication per iteration. In case we are able to store the matrix $Q = A^\top A$, following the same reasoning as before, we have that full gradient calculation in the first case costs $\mathcal{O}(n^2)$ and in the second case only $\mathcal{O}(n)$.[1]

Anyway, when we only need to get one component of the gradient per iteration in a BCGD framework, we might first calculate the residual

$$r(x_{k+1}) = A\left(x_k - g_{i_k} e_{i_k}\right) - b = r(x_k) - g_{i_k} A_{i_k}$$

[1] Obviously storage of Q does not make sense in the full gradient case.

and, keeping in mind that

$$\nabla f(x_{k+1}) = 2A^\top r(x_{k+1}),$$

we get

$$\nabla_{i_k} f(x_{k+1}) = 2a_{i_k}^\top r(x_{k+1}).$$

This has a cost $\mathcal{O}(m)$.[2]

4. (8 POINTS) Consider the problem:

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\|x\|^2 + \max_{i\in I}\{a_i + c_i^\top x\}$$

where $I = \{1,\dots,m\}$, $a_i \in \mathbb{R}$ and $c_i \in \mathbb{R}^n$. Describe the Lagrangian dual of the considered problem and give a possible primal solution.

Solution 4. We can write our problem as follows:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \frac{1}{2}\|x\|^2 + v \\ \text{s.t.} & a_i + c_i^\top x \le v, \quad i \in I \end{array}$$

If we call

$$C = \begin{bmatrix} c_1^\top \\ \dots \\ c_m^\top \end{bmatrix}$$

and a the vector of components a_i, we can write the problem as follows

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \frac{1}{2}\begin{bmatrix} x \\ v \end{bmatrix}^\top \begin{bmatrix} I & 0_n \\ 0_n^\top & 0 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0_n^\top & 1 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix} \\ \text{s.t.} & a + [C - e]\begin{bmatrix} x \\ v \end{bmatrix} \le 0 \end{array}$$

Since we are in the quadratic case described in our notes (Section 5.7.2), we can write the dual as follows

$$\begin{array}{ll} \max\limits_{x\in\mathbb{R}^n} & -\frac{1}{2}\begin{bmatrix} x \\ v \end{bmatrix}^\top \begin{bmatrix} I & 0_n \\ 0_n^\top & 0 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix} + \lambda^\top a \\ \text{s.t.} & e^\top \lambda = 1 \\ & \lambda \ge 0 \end{array}$$

Again taking into account results in Section 5.7.2, we have that

$$x^\star = -C^\top \lambda^* = -\sum_{i=1}^m c_i \lambda_i^*$$

5. (8 POINTS) Consider the problem

$$\max_{x\in P} f(x),$$

with $P \subseteq \mathbb{R}^n$ non-empty polytope and $f(x)$ continuously differentiable convex function. Consider the Frank-Wolfe variant described in Algorithm 1. Prove

Algorithm 1 `Frank-Wolfe for maximizing a convex function over a polytope`

```
1 Choose a point $x_1 \in P$
2 For $k = 1, \dots$
3     Set $\hat{x}_k = \arg\max_{x \in P} \nabla f(x_k)^\top (x - x_k)$
4     If $\nabla f(x_k)^\top (\hat{x}_k - x_k) = 0$, then STOP
5     Set $x_{k+1} = \hat{x}_k$
6 End for
```

that the algorithm converges in a finite number of iterations to a point satisfying optimality conditions[3].

Solution 5. At each iteration we can have two cases. Either

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) = 0,$$

and using the definition of $\hat{x}_k$, we have

$$0 = \nabla f(x_k)^\top (\hat{x}_k - x_k) \geq \nabla f(x_k)^\top (x - x_k) \quad \forall\ x \in P,$$

thus optimality conditions are satisfied (no feasible ascent direction can be obtained) and the algorithm stops with a stationary point, or

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) > 0,$$

and $d_k = \hat{x}_k - x_k$ is an ascent direction. In the second case, using convexity, we can write

$$f(x_{k+1}) = f(\hat{x}_k) \geq f(x_k) + \nabla f(x_k)^\top (\hat{x}_k - x_k) > f(x_k),$$

Thus we get a new point that strictly increases the objective function value. We can hence conclude that the algorithm cannot visit any point twice. Since we move from one vertex to another, the number of vertices is finite, and we have that at least one vertex is a global solution of the problem, the algorithm will stop at a stationary point in a finite number of iterations.

[2] The idea here is keeping track of the residual and calculating the component that is really needed.

[3] Keep in mind that for the considered problem a result similar to the fundamental theorem of linear programming holds. Indeed, at least one of vertex of the polytope is a global maximizer for f over P.