Lecture 15

Waldemar Hebisch

June 14, 2023

1 Proximal methods

Example: Let $h(s) = \|s\|_*$ (nuclear norm of s). On space of square n by n matices we consider euclidean distance in terms of coordinates (which is called Hilbert-Schmidt norm). Both nuclear norm and Hilbert-Schmidt norm does not change when we multiply s from the left or from the right by an orthogonal matrix. So it is enough to compute proximal operator when s diagonal matrix. Morover,

$$\mathrm{argmin}_y(\frac{1}{2}\|y - s\|_2^2 + \|y\|_*$$

it is enough to consider diagonal y. Namely, nuclear norm of orthogonal projection of y onto diagonal matrices is less or equal to nuclear norm of y, so orthogonal projection of y onto diagonal matrices can not increase any of term above.

For diagonal matrices nuclear norm is just sum of absolute values of entries on diagonal, that is nuclear norm equals L^1 of vector formed from diagonal entries. So, we get the following algorithm to compute $\mathrm{prox}_{\alpha h}(s)$ where h is nuclear norm:

- write $s = UDV$ where U and V are orthogonal and D is diagonal (SVD decomposition),
- form vector t from diagonal entries of D,
- compute $r = \mathrm{prox}_{\alpha h_1}(t)$ where h_1 is L^1 norm,
- form diagonal matrix p from entires of r,
- compute UpV.

The following lemma helps with computing projections onto intersection of sets:

Lemma 1.1 *Assume that C_1 and C_2 are closed convex sets, and let $F = C_1 \cap C_2$. If $y = \mathrm{Proj}_{C_1}(x) \in C_2$, then $\mathrm{Proj}_F(x) = y$.*

Proof: Since $F \subset C_1$ we have

$$\min_{z\in F} \frac{1}{2}\|z-x\|^2 \geq \min_{z\in C_1} \frac{1}{2}\|z-x\|^2.$$

By assumption

$$\min_{z\in C_1} \frac{1}{2}\|z-x\|^2 = \frac{1}{2}\|y-x\|^2$$

and since $y \in C_2$ we also have $y \in F$, so

$$\frac{1}{2}\|y-x\|^2 \geq \min_{z\in F} \frac{1}{2}\|z-x\|^2.$$

Together this means

$$\min_{z\in F} \frac{1}{2}\|z-x\|^2 = \min_{z\in C_1} \frac{1}{2}\|z-x\|^2.$$

But $\|z-x\|^2$ is strictly convex, so minimum is attained is single point, so also

$$\mathrm{Proj}_F(x) = \mathrm{argmin}_{z\in F} \frac{1}{2}\|z-x\|^2 = \mathrm{argmin}_{z\in C_1} \frac{1}{2}\|z-x\|^2 = \mathrm{Proj}_{C_1}(x).$$

□

Example: Consider half of unit ball, that is subset of $B = \{x : \|x\| \leq 1\}$ consisting of elements with $x_1 \geq 0$. More precisely, we consider intersection F of B with $H = \{x : x_1 \geq 0\}$. When $x \in H$ it is easy to see that also $\mathrm{Proj}_B(x) \in H$, so by the lemma $\mathrm{Proj}_F(x) = \mathrm{Proj}_B(x)$. When $x \notin H$ we get projection onto F by first projecting onto $P = \{x : x_1 = 0\}$ and then projecting onto $F \cap P$ which is ball in P.

1.1 Acceleration

We can improve convergence using variant of Nesterov acceleration. Put $x_{-1} = x_0$ and

$$v_i = x_i + \frac{i-1}{i+2}(x_i - x_{i-1})$$
$$x_{i+1} = \mathrm{prox}_{\alpha_i h}(v_i - \alpha_i \nabla g(v_i))$$

First two steps are just usual proximal proximal gradient steps, other contain momentum term.

Accelerated proximal gradient algorithm applied to LASSO problem is called FISTA

1.2 Proximal Newton method

Simply applying proximal operator to Newton step does not work. But if we recall that Newton method is equvalent to steepest descent with rescaled metric, we can get correct proxmal variant of Newton method: we need to compute proximal operator with respect to rescaled metric. Unfortunately, this is usually quite expensive so this is of limited use.

2 Barrier methods

Recall constrained optimization:

$$\min_{x \in S} f(x)$$

Instead of minimizing f in barrier methods we optimize

$$f(x) + \lambda h(x)$$

where $h(x) \geq 0$ and $h(x)$ goes to infinity when x goes to boundary of S for sequence of $\lambda > 0$ going to 0. h above is called barrier function. In pure barrier methods S must be closure of its interior, so we do not allow equality constraints (they must be handled separately).

Classicaly $S = \{x : \forall_{i=1,\ldots,N} g_i(x) < 0\}$. In such case popular choice of barrier function h are

$$h(x) = \sum_{i \in I} \frac{1}{g_i^2(x)}$$

and

$$h(x) = -\sum_{i \in I} \log(-g_i(x)).$$

There are advantages of use of self-concordant barrier functions. For many important convex S self-concordant barrier functions are easy to construct.

When using barrier function we can use unconstained method to optimize $f(x) + \lambda h(x)$, in particular we can use Newton method.

Correctness of of barrier methods follows from the following lemma

Lemma 2.1 *If f is continous, h is finite in interior of S, S is equal to closure of its interior,*

$$x_\lambda = \operatorname{argmin}_{x \in S}(f(x) + \lambda h(x)),$$

$$x_\lambda \to x_\infty,$$

then

$$x_\infty = \operatorname{argmin}_{x \in S} f(x).$$

Proof: Fix $\varepsilon > 0$. By definition of inf there exists $y \in S$ such that

$$f(y) \leq \inf_{x \in S} f(x) + \varepsilon.$$

y may be on boundary of S, but since f is continuous and S is closure of its interor there exists z in interior of S such that

$$f(z) \leq \inf_{x \in S} f(x) + 2\varepsilon.$$

Since $h(z)$ is finite for small λ we have

$$f(z) + \lambda h(z) \leq \inf_{x \in S} f(x) + 3\varepsilon$$

so

$$f(x_\lambda) \le f(x_\lambda) + \lambda h(x_\lambda) = \min_{x \in S}(f(x) + \lambda h(x))$$
$$\le f(z) + \lambda h(z) \le \inf_{x \in S} f(x) + 3\varepsilon.$$

Since $x_\lambda \to x_\infty$ and f is continous also

$$f(x_\infty) \le \inf_{x \in S} f(x) + 3\varepsilon.$$

Since $\varepsilon > 0$ were arbitrary we have

$$f(x_\infty) \le \inf_{x \in S} f(x)$$

so

$$f(x_\infty) = \min_{x \in S} f(x).$$

□

Features of barrier methods:

- need feasible starting point
- goes only trough feasible points
- can use efficient unconstrained method like Newton method
- barrier function is typically badly conditioned in classical sense
- self-concordant barriers have very good convergence properties with Newton method

Interior point method for linear programming is an example of barrier method. In case of linerar programming crucial point is that goal function is linear (affine) and

$$h(h) = -\sum \log(x_i)$$

is self-concordant. However, several problems may be converted to form with linear goal function on set S having self-concordant goal function.

Example: In semidefinite programming goal function is linear and constraints have form $g_i(x) \in P_i$ where g_i is affine function and P_i is cone of positive definite matrices (we use index i because dinension of matrices may depend on i). Recall, that on positive definite matrices

$$h(S) = -\log(\det(S))$$

is self-concordant. So each constraint of form $g_i(x) \in P_i$ can be replaced by self-concordant barrier and all such constraints by sum (which again is self-concordant).

Computing such barrier function is more expensive than in case of linear programming (because simple way to compute barrier function need to diagonalize corresponding matrices) and some specific optimizations for linear programming do not apply. By we can solve semidefinite problems by method which is essentially the same as interior point method for linear programming.

Semidefinite programming may look special, but in fact some popular problems are special cases of semidefinite programming:

- quadratic problems: constraints are affine, goal function is convex quadratic
- quadratically constrained quadratic problems: constraints and goal function are convex quadratic
- second order cone problems

Clearly, quadratically constrained quadratic problems are more general than quadratic problems.

Second order cone problems have affine goal function and constraints of form

$$\|Ax + b\| \leq \langle c, x\rangle + d.$$

If needed adding extra variables we can transform convex quadratic constraints into second order cone constraints. Also, by adding extra variable t and constraint $f - t \leq 0$ we can transform problem into problem with linear goal function. So in this sense second order cone problems are more general than quadratically constrained quadratic problems.

Note that

$$\|x\| \leq t$$

is equivalent to

$$0 \leq \begin{pmatrix} tI & x \\ x^T & t \end{pmatrix}$$

so we can rewrite second order cone constraints as semi-definite constraints.

References

Literature about interior point methods is quite extensive, most of it is also quite advanced. Resonable introduction is given in Nemirovski lectures (linked from course web page).

3 Practical aspects

3.1 3-SAT

In the first lecture we had the following example. Let

$$s_1 = x_1 + x_2 + x_3,$$

$$s_2 = x_1x_2 + x_1x_3 + x_2x_3,$$

$$s_3 = x_1x_2x_3,$$

$$Q = 1 - (s_1^2 - 3s_2 + s_3).$$

One can check that for $x \in [0,1]^3$ we have $0 \le Q \le 1$ with equality only at vertices of the cube. Moreover $Q(0,0,0) = 1$ and at other vertices $Q = 0$.

Consider boolean formulas in variables $x_1, \ldots, x_k$. Let x_{i+k} be negation of x_i (this is to avoid explicitly writing negations). Given boolean formula

$$B = (x_{j_{1,1}} \vee x_{j_{2,1}} \vee x_{j_{3,1}}) \wedge \cdots \wedge (x_{j_{1,m}} \vee x_{j_{2,m}} \vee x_{j_{3,m}})$$

we build polynomial f in $y_1, \ldots, y_k$ as

$$f = Q(y_{j_{1,1}}, y_{j_{2,1}}, y_{j_{3,1}}) + \cdots + Q(y_{j_{1,m}}, y_{j_{2,m}}, y_{j_{3,m}})$$

where $y_{i+k} = 1 - y_i$. Let S be unit hypercube, that is set of y such that $0 \le y_i \le 1$ for $i = 1, \ldots, k$. One can show that $\min f$ on S is zero if and only if there is substitution of truth values for variables in B which makes B true.

Discrete problem above is usually called 3-SAT. It is NP-complete problem. However, random instances of 3-SAT with $m < 3k$ tend to be easily solvable. So it makes sense to check how our methods work.

Trying projected gradient descent with learning rate $\alpha = 0.3$ and with random starting point in $[0,1]^k$ converges to local minimum in relatively small number of iterations: of order 7 when $k = 350$ and $m = 820$ growing to about 20 when $k = 80000$ and $m = 150000$. But values at local minimum are rather large and even large number of trials like 2000 did not give 0 as minimal value.

When we want to solve 3-SAT we can try differen function:

$$P = 1 - (s1 - s2 + s3).$$

One can show that P for $x \in [0,1]^3$ satisfies $0 \le P \le 1$. Morover $P(0,0,0) = 1$ and at over vertices $P = 0$. So instead of Q we could try P.

This time convergence needs more iterations, of order 30 when $k = 350$ and $m = 820$ and of order 150 when $k = 80000$ and $m = 150000$. But when $k = 350$ and $m = 820$ in moderate number of trials we get value 0 (that is satisfying substitution). Unfortunately, when $k = 80000$ and $m = 150000$ we get relatively small values but probablity of getting 0 seem to be quite small.

The full program is example `sat3.c`, below is projected gradient part:

```
float
pgd(struct form * f, float * x, float alpha, int N) {
    int m = f->nv;
    float * dx = my_malloc(m*sizeof(*dx));
    int l;
    float res;
    for (l=0;l < N; l++) {
        res = eval_form(f, x, dx);
        int i;
        float del2 = 0;
        for(i = 0; i < m; i++) {
            float nx = x[i] - alpha*dx[i];
            /* Compute projection */
            nx = (nx < 0)?0:nx;
            nx = (nx > 1)?1:nx;
            /* Add contribution to norm of projected gradient */
            float di = x[i] - nx;
            del2 += di*di;
            x[i] = nx;
        }
        if (del2 < 0.00001) {
            break;
        }
    return res;
}
```

3.2 Beck and Teboulle method

Beck and Teboulle method is releated to ADMM and proximal methods. We want to compute

$$\operatorname{argmin}_x \|Ax - b\|^2 + 2\lambda\|x\|_{TV}$$

where $\|\cdot\|$ is l^2-type norm and $\|\cdot\|_{TV}$ is discrete total-variation seminorm, x is image to be recovered, b is observed noisy image and A is transformation matrix representing blurring (in simplest case $A = I$ is identity matrix). For infinite grayscale images

$$\|x\|_{TV} = \sum_{i,j} |x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|.$$

For finite images we skip differences when one of elements is outside. For color images we can compute total variation separately for each color component (as done in example program), or (better) treat each pixel as vector and compute euclidian norm of difference.

Like in ADMM Beck and Teboulle use duality. However, since dual problem is strongly conves they use accelerated projected gradient method to solve the dual problem.

Their method converges in small number of iterations. Due to non-smoothness and high dimensions methods like gradient descent are likely to converge very slowly (and the same for conjugate gradient). ADMM is likely to work well too, but Beck and Teboulle method has small per-iteration cost, so it is hard to beat.