Practical aspects

Waldemar Hebisch

January 25, 2022

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 $0 \leq Q \leq 1$ 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 $0 \leq y_i \leq 1$ 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 nu,mber of iteration: 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 evan 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 $0 \leq P \leq 1$. 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 exmaple `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;
}
```

2 Boyd and Vanderberge interior point examples

Boyd and Vanderberge on page 614 and 615 show their result: with $\mu = 10$ interior point methods for linear programming for moderate dimension (100) needs less than 30 iterations and for larger dimensions number of iterations grow slowly, up to about 35 for dimension 1000.

3 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.