Active set methods

Anton Ruby Larsen [s174356]

December 13, 2021

1 Primal active set method

The theory is described in this source. Also a pseudo-code formulation of the algorithm can be find in the given source. We have implemented the method here

2 Dual active set method

We will in this section derive the dual active set method from our primal problem formulation. The derivation is based on [1]. Our primal problem is stated in the form

$$\begin{array}{ll} \min_{x\in R^n} & f(x) = \frac{1}{2}x^T Hx + g^T x \\ \text{s.t.} & c_i(x) = a_i^T x + b_i \geq 0 \quad i \in \mathcal{I} \end{array} \tag{1}$$

where H must be strictly convex for the dual method. The dual problem is then given as

$$\begin{array}{ll} \max\limits_{x,\lambda} & \mathcal{L}(x,\lambda) = \dfrac{1}{2}x^T Hx + g^T x - \lambda^T(A^T x - b) \\ \text{s.t.} & \nabla_x \mathcal{L}(x,\lambda) = Hx + g - A\lambda = 0 \end{array} \tag{2}$$

As seen in exercise 1.9 we know from section 2.7 in [2] that the KKT conditions are both necessary and sufficient for both the primal and the dual because we have restricted H to be strictly convex and all the c_is are affine. We now introduce a primal working set, W, and a dual working set, W_D. The primal working set comply with

$$\begin{aligned} Hx^{(k)} + g - \sum_{i\in W} a_i \lambda_i^{(k)} &= 0 \\ c_i(x^{(k)}) = a_i^T x^{(k)} - b_i &= 0, \quad i \in W \\ \lambda_i^{(k)} &\geq 0, \quad i \in W. \end{aligned}$$

And the dual set follows

$$\begin{aligned} Hx^{(k)} + g - \sum_{i\in W_D} a_i \lambda_i^{(k)} &= 0 \\ \lambda_i^{(k)} &= 0, \quad i \in W_D. \end{aligned}$$

We can now from 1, 2 and the KKT conditions infer that x is optimal when

$$c_i(x^{(k)}) \geq 0, \forall i \in W_D$$

We further observe that when ever there exists a constraint, c_r, not satisfying this in the dual set we are not optimal. This observation we will use to develop a iterative approach.

2.1 Improving direction and step length

We know that if some constraint c_r is negative we can increase the corresponding λ_r which will improve our dual problem 2 without causing it getting infeasible. This can formally be written as

$$\begin{aligned} \bar{x} &= x + s \\ \bar{\lambda}_i &= \lambda_i + u_i, \quad i \in W \\ \bar{\lambda}_r &= \lambda_r + t \\ \bar{\lambda}_i &= 0 \quad i \in W_D n\{r\} \end{aligned} \tag{3}$$

Using the KKT conditions we can formulate this similarly to the KKT systems we have seen earlier.

$$\begin{bmatrix} H & -A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} + \begin{bmatrix} g \\ b \end{bmatrix} - \begin{bmatrix} a_r \\ 0 \end{bmatrix} \lambda_r + \\ \begin{bmatrix} H & -A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} s \\ u \end{bmatrix} - \begin{bmatrix} a_r \\ 0 \end{bmatrix} t = 0. \tag{4}$$

Using the KKT conditions again we also know that the first line of 4 must equal 0 so we can reduce it to

$$\begin{bmatrix} H & -A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} s \\ u \end{bmatrix} - \begin{bmatrix} a_r \\ 0 \end{bmatrix} t = 0. \tag{5}$$

We now 5 to

$$\begin{bmatrix} H & -A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} a_r \\ 0 \end{bmatrix}, \quad \text{where} \quad \begin{bmatrix} s \\ u \end{bmatrix} = \begin{bmatrix} p \\ v \end{bmatrix} t$$

We have now the primal space direction, p, the dual space direction v and the step length t. We can solve the above system for the directions but the step length t we need to do a little more work to obtain a method which we can use to calculate it. We rewrite 3.

$$\begin{aligned} \bar{x} &= x + s = x + pt \\ \bar{\lambda}_i &= \lambda_i + u_i = \lambda_i + v_i t, \quad i \in W \\ \bar{\lambda}_r &= \lambda_r + t \\ \bar{\lambda}_i &= 0 \quad i \in W_D n\{r\} \end{aligned} \tag{6}$$

We know all λ's must be non-negative for 2 to be feasible. Therefore we infer that it must hold for condition 2 in 6 that

$$t \in [0, t_{max}], \text{ where } \quad t_{max} = min(\infty, min_{i:v_i<0} \tfrac{\lambda_i}{v_i}),\ i \in W_D$$

We now have a method to derive a maximal step length in the dual space and we know from [1] that all step sizes $t > 0$ increases c_r so we do not have to worry about the primal problem. We will therefore now derive the optimal step length only using 2. We know from [1] that

$$\mathcal{L}(\bar{x}, \bar{\lambda}) = \mathcal{L}(x, \lambda) - \frac{1}{2} t^2 a_r^T p - t c_r(x) \Rightarrow$$
$$\mathcal{L}(\bar{x}, \bar{\lambda}) - \mathcal{L}(x, \lambda) = -\frac{1}{2} t^2 a_r^T p - t c_r(x)$$

We set the r.h.s. equal to zero and differentiate w.r.t. t to maximize the increase in $\mathcal{L}(\bar{x}, \bar{\lambda})$.

$$\frac{d\mathcal{L}}{dt} = -t a_r^T p - c_r(x) \Rightarrow t^* = \frac{-c_r(x)}{a_r^T p}$$

We now know that t must equal

$$t = min(t_{max}, t^*)$$

for both maximizing 2 and remaining feasible.

Lastly before implementing the algorithm we must make sure $A = [a_i]_{i \in W}$ atains full rank after adding c_r. From [1] we know that when

$$a_r^T p = 0$$

Then c_r and $A = [a_i]_{i \in W}$ are linearly dependent. To solve this we remove c_j from W where j equals

$$\arg\min_{j:v_j<0} \frac{-\lambda_j}{v_j}$$

If we are not optimal and no $v_j < 0$ exists the problem is infeasible.

2.2 Implementation

The dual active-set method has two advantages over the primal formulation. First we do not have issues regarding cycling and secondly we do not need to solve a phase-1 problem to obtain an initial starting point and working set. We can just solve the unconstrained problem

$$\begin{aligned} x_0 &= -H^{-1}g \\ W_0 &= \{\} \\ W_D 0 &= \{c_i\}, \quad \forall i \in \mathcal{I} \end{aligned}$$

If one wants to see a pseudo-code formulation of the algorithm we recommend this pseudo-code formulation. A implementation of the method in matlab is available here

3 Revised Simplex

The following derivation is based on lecture 11 and 13 in the course [3]. The derivation will start with a standard form LP.

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Now assume that A is full rank and the rows of A are linearly independent.

$$A \in R^{m \times n} \quad rank(A) = m \quad m \leq n$$

Assume further that the variables have been written in such an order that the last m columns of A are linearly independent. We can then partition A and x as follows

$$\begin{aligned} A &= \begin{bmatrix} A_N & A_B \end{bmatrix}, \quad A_B \in R^{m \times m}, \; A_N \in R^{m \times (n-m)} \\ x &= \begin{bmatrix} x_N \\ x_B \end{bmatrix}, \quad x_B \in R^m, \; x_N \in R^{n-m} \end{aligned} \tag{7}$$

The constraints $Ax = b$ have now become

$$\begin{aligned} \begin{bmatrix} A_N & A_B \end{bmatrix} \begin{bmatrix} x_N \\ x_B \end{bmatrix} &= b \Rightarrow \\ A_N x_N + A_B x_B &= b \end{aligned} \tag{8}$$

If we set $x_N = 0$ then (8) becomes

$$\begin{aligned} A_B x_B &= b \Rightarrow \\ x_B &= A_B^{-1} b \end{aligned} \tag{9}$$

The resulting $x = \begin{bmatrix} x_N \\ x_B \end{bmatrix}$ has at least $n - m$ zero elements and is called a basic solution.
x_N are the non-basic variables and x_B are the basic variables.

From the Fundamental theorem of linear programming we know that if A is assumed full rank then

1. If a feasible solution exists then a basic feasible solution(BFS) exists
2. If an optimal feasible solution exists then a optimal BFS exists

So this theorem says we only need to consider basic feasible solutions when searching for an optimum, i.e. we are looking for an optimal basis B.
The simplex method then says: Once we have a basic feasible solution we try to improve it by replacing one basic variable with a non-basic variable. We continue until no improvement to $c^T x$ is possible.
Assume we have a BFS $x^{(k)}$ with sets B and N of basic and non-basic variables. We compute

$$x_B^{(k)} = A_B^{-1} b$$

and the cost is

$$\begin{aligned} c^T x^{(k)} &= c_N^T x_N^{(k)} + c_B x_B^{(k)} \\ &= c_B^T x_B^{(k)} \end{aligned}$$

We want to see what happens to the cost if we start changing the non-basic variables from zero. If we change x_N, we need to recompute x_B from (8).

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N$$

This is possible because A_B is invertible by design. The cost can then be expressed in terms of x_N as

$$\begin{aligned} c^T x^{(k)} &= c_B^T x_B^{(k)} + c_N^T x_N^{(k)} \\ &= c_B^T A_B^{-1} b + (c_N^T x_N^{(k)} - c_B^T A_B^{-1} A_N) x_N^{(k)} \\ &= c^T x^{(k)} + r^T x_N^{(k)} \end{aligned}$$

where $r = c_N - A_N^T A_B^{-T} c_B$. This vector r is known as the vector of reduced cost or the Lagrange multipliers. r quantifies how much each non-basic variable will affect the cost if increased, i.e. if entry 1 of r is positive $x_N[1]$ will increase the cost and vice versa.
Therefore if $r \succeq 0$ then no improvement is possible which means we are optimal. Otherwise we pick the most negative entry of r and increase the corresponding non-basic variable, x_e, called the entering variable.
We now need to find which variable in the basic set which have to leave. To find this we will start with

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N$$

Which becomes

$$\begin{aligned} x_B &= A_B^{-1} b - A_B^{-1} a_e x_e \\ &= \hat{b} + d x_e \end{aligned}$$

where a_e is the column of A corresponding to x_e.
The variable x_i, $i \in B$, becomes zero when

$$x_e = \frac{\hat{b}_i}{-d_i}$$

We are interested in the first basic variable, x_l, to reach zero

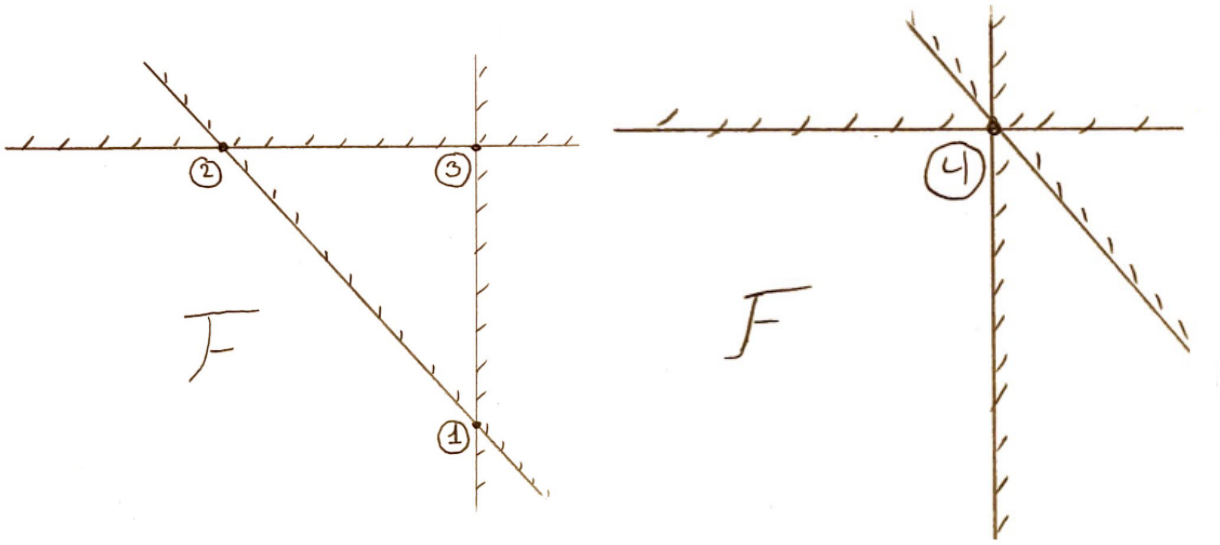
$$\frac{\hat{b}_l}{-d_l} = \min_{i \in B : d_i < 0} \frac{\hat{b}_i}{-d_i}$$

If no $d_i < 0$ then we know x_e can be increased indefinitely which means the LP is unbounded. We now switch l and e in the sets N and B. We then update A_N and A_B and proceed to the next iteration. It can be shown that our new A_B is of full rank so invertibility is not a problem.

It may be that one $\hat{b}_i = 0$ and $d_i < 0$. In this case x_e can not be increased, so the objective is not decreased. We can still switch the entering and the leaving variable to get a new basis, but our objective remain the same. In this case it is possible to return to the same basis set and begin cycling. This phenomenon is called degeneracy which active set-methods some times are vulnerable to as we also saw for the primal active-set method for IQPs.

3.1 Degeneracy

A BFS is called degenerate if $x_i = 0$ for at least one i in the basis. Geometrically degeneracy is when two basic solutions overlap.



(a) Not degenerate

(b) Degenerate

Figure 1: We see that the vertical constraint has been shifted to the left from figure (a) to figure (b) which has made (b) degenerate.

We see that the three basic solutions in 1a have merged into one in 1b. This phenomenon can as mentioned cause cycling which is when a sequence of bases $B_1 \dots B_k$ occurs repeatedly. This requires that

1. Each basis $B_1 \dots B_k$ is degenerate as the entering variable is 0
2. No x_e is altered so all BFSs are the same point as in 1b

An example of a LP which will cause the plain vanilla simplex to cycle is Beal's example given in section 4 of this paper.

A solution to the cycling problem is Bland's anti-cycling rules

1. Of all non-basic variables for which the corresponding entry of r is negative, choose the one with the smallest index to be the entering variable.
2. If there is a tie for the leaving variable, pick the one with the smallest index.

These rules only needs to be invoked if no decrease in $c^T x$ is obtained.

3.2 Implementation

The starting assumption of the revised simplex algorithm is that we have a BFS. This is called the phase-1 problem and is almost as much work as solving the LP it self. We will not dive further into the topic here but only recommend this source where three different methods are explained.

A pseudo-code for the revised simplex algorithm is stated below.

Revised Simplex

1. Initialization: We have sets B_0, N_0 and corresponding A_B and $x_B = A_B^{-1}b$
2. Iterate until $r \succeq 0$
   (a) Compute reduced costs: $r = c_N - A_N^T A_B^{-T} c_B$
   (b) We find the index e of r which is the most negative. This x_e from the set N is our entering variable.
   (c) We now compute the leaving variable

$$\hat{b} = A_B^{-1}b, \quad d = -A_B^{-1}a_e$$

$$l = \underset{i \in B: d_i < 0}{\arg\min} \frac{\hat{b}_i}{-d_i}$$

   where x_l from the set B is our leaving variable.
   (d) If any of the ratios $\frac{\hat{b}_i}{-d_i}$ are 0 we restart the iterate with Bland's rules invoked.
   (e) We update N and B

We see that we in the above algorithm is calculating A_B^{-1} so when implementing the algorithm we should calculate the factorization every iterate and then reuse it for all three inverse. An even faster implementation would make use of something as a rank-1 update of the inverse every iteration because we are only changing one row per iteration. We will not implement this in our implementation but if one wants to know more about this topic we recommend this source.

We have implemented a basic version of the Simplex method here

References

[1] E. L. Hansen and C. Völcker, "Numerical algorithms for sequential quadratic optimization," 2007.

[2] J. B. Jørgensen, *Numerical Methods for Constrained Optimization*. Springer, first ed., 2021.

[3] D. R. Brown, "Uc math 352, university of canterbury." https://www.youtube.com/playlist?list=PLh464gFUoJWOmBYla3zbZbc4nv2AXez6X.