Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 53

Hello everyone, this is the third lecture of week 11. In the previous lecture, we started with Karmarkar's algorithm. We learnt how to convert a given problem to the canonical form and we also learnt what a projective transformation is. Recalling that, you can look at the screen and note what the canonical form and projective transformation are. This means we are going to solve a problem which is in this form:

minimize $c^T x$

subject to

$A x = 0$, $a^T x = 1$, and $x \geq 0$.

If you apply the projective transformation, it says that

$y_k = (X_k^{-1} x_k) / (1^T X_k^{-1} x_k)$ and the $x_k = (X_k y_k) / (a^T X_k y_k)$.

You can actually see that any point x_k is converted to $(1/n, 1/n, ..., 1/n)$.

That is because X_k is just the diagonal matrix of x_k.

So $X_k^{-1} X_k$ will give you all ones, and 1^T of all ones is n.

So you will have y_k will be $(1/n, 1/n, ..., 1/n)$, which is $(1/n)\, 1$.

So that is where you transform x_k to. In the affine scaling method, from x_k we transform to all ones. Here we are transforming x_k to all $(1/n, 1/n, ..., 1/n)$.

You will always have $1^T y = 1$ because of that.

The equivalent constraint of $a^T x = 1$ is actually $1^T y = 1$.

The transformed problem after the projective transformation is the following:

minimize $c^T X_k y_k / (a^T X_k y_k)$.

I have just substituted x as x_k, so $x_k = X_k y_k / (a^T X_k y_k)$, subject to the conditions that A multiplied by this substitution, but note that the denominator does not matter because the right-hand side is 0.

So this just becomes $A X_k y_k = 0$, and $a^T x$ becomes $1^T y$,

so this gets converted to $1^T y = 1$, and $y \geq 0$.

The transformed problem appears very similar, but there is actually a very big problem that some of you might have figured out. You can see that the problem is no longer a linear programming problem.

Why is that? Because if you see the objective function, y is in the denominator. This is a minimization with respect to y_k, which is linear in the numerator and linear in the denominator. This problem is no longer a linear programming problem, but we want it to be a linear programming problem.

That is the reason why the optimal value had to be 0. You might have wondered why we had this condition that the optimal value of c^Tx, c^Tx^*, must be equal to 0.

It was because if the answer is 0, you can actually remove the denominator. You can transform this as

minimize $c^T X_k y_k$

subject to

$A X_k y_k = 0$, $1^Ty = 1$, and $y \geq 0$.

Because if the optimal answer is 0, that means $c^T X_k y_k$ itself is 0. It cannot be otherwise. That is the reason we want the optimal answer to be equal to 0.

We will see how to get around this. We will continue with this problem that you are minimizing $c^T X_k y_k$ subject to $A X_k y_k = 0$, $1^Ty = 1$, $y \geq 0$.

One thing you can observe now is that the transformed problem is very similar to what you got in the affine scaling method. There also the transformed problem was

minimize $c^T X_k y_k$

subject to

$A X_k y_k = 0$ and $y \geq 0$.

You do not have this extra constraint $1^Ty = 1$ here.

But the projection was, if you could recall, so you can observe that the transformed problem is very similar to what you got in the affine scaling method. If you recall this step, this was what you had initially, and after doing the linear transformation you got

minimize $c^T X y$

subject to

$A X y = b$, $y \geq 0$.

That is when we derived the big projection that you have here which we converted to

$X (A^T\mu - c)$.

This is very similar to what you have here where you are minimizing $c^T X y$

subject to $A X y = 0$,

but you have this extra constraint $1^T y = 1$.

That is the only difference, but otherwise the transformed problem is exactly the same as what you had for the affine scaling method.

Let us go ahead with this. What we have done until now is that we have converted the problem to canonical form, used the projective transformation, and have converged to this particular problem now.

Starting at some point x_0, we have to find a descent direction.

Now we starting at x_0, we find the steepest descent direction, which is $-\nabla f = -c$, projected onto the feasible set, the feasible polytope.

If you recall the projection of $-c$ onto the null space of A is $-[I - A^T (A A^T)^{-1} A] c$, so it is actually this multiplied by $-c$, but both are the same.

Following the same thing here, the projection of $-X c$ because the gradient in the transformed space is $X c$, onto the null space of $A X$, is $-[I - X A^T (A X^2 A^T)^{-1} A X] X c$.

This is actually the same as $-X c$ and this part is μ, so $X A^T \mu$, so that is - of or rather to take the minus in it is $A^T\mu - c$.

That is the projection of $-X c$ into the null space of $A X$.

It is all the same as what we did in the affine scaling method, but we have to do one more thing because we also have to project this to $1^T y = 1$.

What I am doing is that I am first projecting onto the space $A X_k y_k = 0$ and then I am projecting again onto 1^T.

I should have written $1 y_k$ since I have written y_k everywhere. I am now projecting it onto this matrix 1. You can say it is a vector, but it is also a matrix of size $1 \times n$.

Consider this as a matrix. Now projection of this x into, maybe I will write that down, what is the problem? Projection of x into $(A^T\mu - c)$ to the null space of 1^T.

This is the matrix whose null space we are projecting this onto. What should this be? This should be $[I - 1 (1^T 1)^{-1} 1^T] X (A^T\mu - c)$.

But what is $1^T 1$? That is actually n, because 1^T is a row vector of $[1, 1, ..., 1]$, and 1 is the column vector of 1s, $[1, 1, ..., 1]^T$. So $1^T 1$ is actually n.

Use that here. So you will just get $[I - (1/n) 1 1^T] X (A^T\mu - c)$.

This is the d_k that we will be using. I possibly had to write X_k and μ_k.

I will just write as d.

If you are wondering what this is, this is actually the steepest descent direction -c which has been projected onto the feasible polytope.

I have gone through quite a few steps because I am first projecting it onto the polytope $A X_k y_k = 0$ and then I am projecting it onto $1^T y_k = 1$.

The expression clearly is just $[I - (1/n) 1 1^T] X (A^T\mu - c)$.

That is the direction. So what we have now done is we first converted it into canonical form, then used projective transformation, and somehow made it a linear program. We have now a transformed linear program in which we have started with an initial point and found the direction to travel. We also need to find the step size.

That derivation I am actually not going to give. The choice of step size is one of the key reasons why Karmarkar could prove that his algorithm converges in a lesser number of steps, in fact polynomial in n.

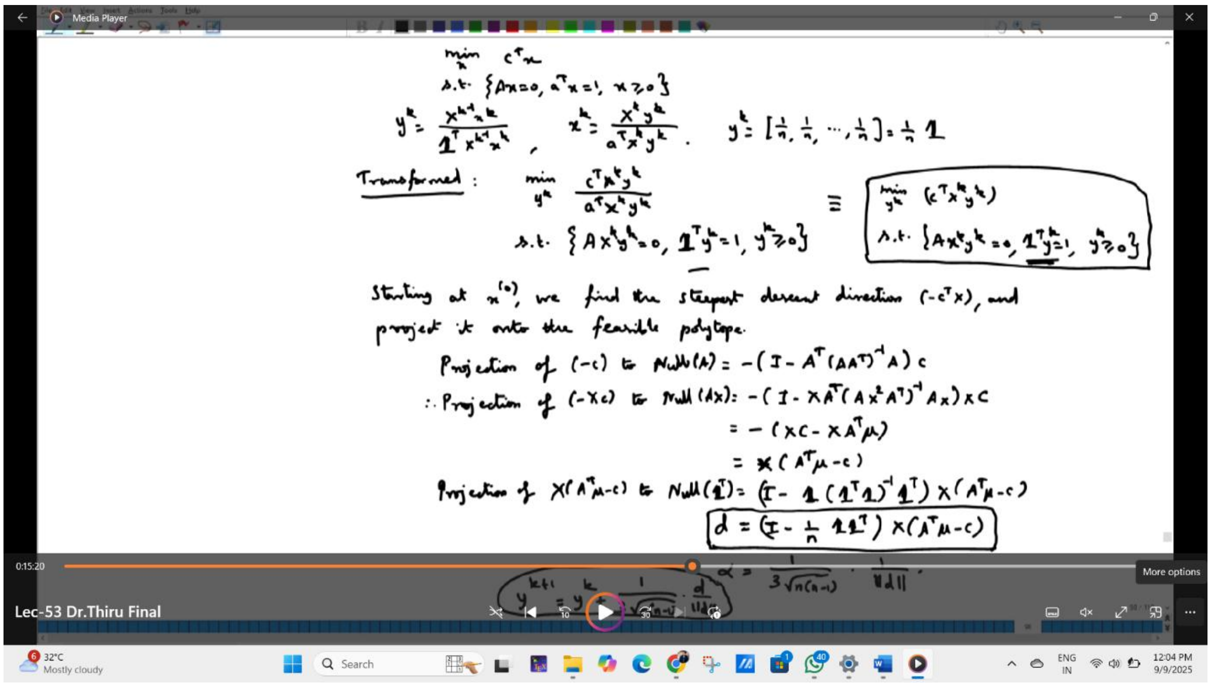
Nevertheless, I will just give the value of α: α will actually be $(1 / (3\sqrt{(n (n - 1))})) (1 / \|d\|)$.

So what you will actually have is

$y_{k+1} = y_k + (1 / (3\sqrt{(n (n - 1))})) (d / \|d\|)$.

This is something we will use in the algorithm, but the derivation of α I have not provided. That is complicated. I am just leaving it; people who are interested can read why such an α is being chosen. As I said, this is a key reason why the algorithm is converging in a lesser number of steps. After this, you do the reverse transformation. Now we will write down the algorithm based on what we have done until now. Let us write down the steps.

(Refer Slide Time 15:20)



$$\min_{x} \; c^T x$$

$$\text{s.t. } \{Ax = 0, \; a^T x = 1, \; x \geq 0\}$$

$$y^k = \frac{X^{k-1} x^k}{\mathbb{1}^T X^{k-1} x^k}, \quad x^k = \frac{X^k y^k}{a^T X^k y^k}. \quad y^k = \left[\tfrac{1}{n}, \tfrac{1}{n}, \cdots, \tfrac{1}{n}\right] = \tfrac{1}{n}\mathbb{1}$$

<u>Transformed</u> :

$$\min_{y^k} \frac{c^T X^k y^k}{a^T X^k y^k} \quad \text{s.t. } \{A X^k y^k = 0, \; \mathbb{1}^T y^k = 1, \; y^k \geq 0\} \equiv \boxed{\min_{y^k} (c^T X^k y^k) \quad \text{s.t. } \{A X^k y^k = 0, \; \mathbb{1}^T y^k = 1, \; y^k \geq 0\}}$$

Starting at $x^{(0)}$, we find the steepest descent direction $(-c^T x)$, and project it onto the feasible polytope.

$$\text{Projection of } (-c) \text{ to } \text{Null}(A) = -(I - A^T (AA^T)^{-1} A)c$$

$$\therefore \text{Projection of } (-Xc) \text{ to } \text{Null}(AX) = -(I - XA^T (AX^2A^T)^{-1} AX)Xc$$

$$= -(Xc - XA^T\mu)$$

$$= X(A^T\mu - c)$$

$$\text{Projection of } X(A^T\mu - c) \text{ to } \text{Null}(\mathbb{1}^T) = (I - \mathbb{1}(\mathbb{1}^T\mathbb{1})^{-1}\mathbb{1}^T)X(A^T\mu - c)$$

$$\boxed{d = (I - \tfrac{1}{n}\mathbb{1}\mathbb{1}^T)X(A^T\mu - c)}$$

$$\alpha = \frac{1}{3\sqrt{n(n-1)}} \cdot \frac{1}{\|d\|}.$$

$$y^{k+1} = y^k + \frac{1}{\sqrt{n(n-1)}} \cdot \frac{d}{\|d\|}$$

We first converted into canonical form.

After this, we did this conversion which is

$y_k = (X_k^{-1} x_k) / (1^T X_k^{-1} x_k)$,

and because of which we got this particular form which is

minimize $c^T X_k y_k$

subject to

$A X_k y_k = 0$, $1^T y_k = 1$, and $y_k \geq 0$.

After this, similar to the affine scaling method,

$\mu_k = (A X_k^2 A^T)^{-1} A X_k^2 c$.

I have not written X_k as the diagonal of x_k, but that is okay.

Now, I can write d_k as $[I - (1/n) 1 1^T] X_k (A \mu_k - c)$, and $y_{k+1} = y_k$, which is $(1/n) 1$, $+ (1 / (3\sqrt{(n (n - 1)}))) (d_k / \|d_k\|)$.

And x_{k+1} is nothing but the inverse transformation, which is $(X_k y_{k+1}) / (a^T X_k y_{k+1})$.

This needs to continue.

Given that you have X_k, now do $k = k + 1$ and continue this.

This needs to go inside a loop.

There is actually an issue regarding the stopping criterion.

Of course, if I just want to write the stopping criterion, it is very easy because we have assumed that the optimal value $c^T x^*$ is 0.

So you can continue this until $|c^T x_k| > \varepsilon$, the tolerance.

If $|c^T x_k| \leq$ tolerance, you can stop because you are converging near the answer which is 0. This is fine, but note that we have actually not solved the issue of not knowing v^*.

Suppose some genie comes and tells us that v^* is the optimal value, then yes, we would have converted it into the canonical form of our wish. The canonical form is of this particular form where v^* is required so that the optimal value actually becomes $c^T x^* = 0$.

But if you do not know v^*, many things actually go berserk. If the value is not v^*, the first thing is the transformed problem is no longer a linear programming problem. It turns out that the transformed problem is a nonlinear programming problem.

Then all of these methods need not work. That is the first problem. The second problem is our stopping criterion also will not work. The stopping criterion is that the modulus of $c^T x_k$ should be going closer to 0.

If it is going closer to the right answer which is, say, 10.57, then we are never going to converge; we are actually going to get into an infinite loop. The stopping criterion is never going to be satisfied. How to get around this? Let us actually note that though we have written $a^T x$ as general as possible, the a^T that we have here, where we convert the given problem to canonical form, the a vector is always n-1 zeros and 1.

That never changes. Since it is an n-dimensional problem and I am introducing n+1, I have written as x_{n+1}. But in general, if you consider the canonical form as an $m \times n$ matrix, with x as an n-dimensional vector, a is actually n-1 zeros and a 1.

What happens here is that the transformed problem becomes

minimize $c^T X_k X_k y_k$, but this is $a^T X_k X_k y_k$.

But a^T as we know is all zeros and one. That is a^T.

So maybe I should change this here as well as a^T.

This into $X_k y_k$.

Now, what is the denominator now? X_k is a matrix of $x_{1,k}$ to $x_{n,k}$ with all zeros, and y_k is y_{1k} to $y_{n,k}$.

So $X_k y_k$ is $x_{1,k} y_{1,k}, x_{2,k} y_{2,k}, \ldots, x_{n,k} y_{n,k}$.

Now, $a^T X_k y_k$ is just $x_{n,k} y_{n,k}$.

So the objective function actually becomes $c^T X_k y_k / (x_{nk} y_{nk})$.

The nonlinearity comes only because of $y_{n,k}$ here. Now when you consider the transformed problem, it is minimize $c^T X_k y_k / (x_{n,k} y_{n,k})$.

Suppose you had, so the c here if you recall, it has -v*.

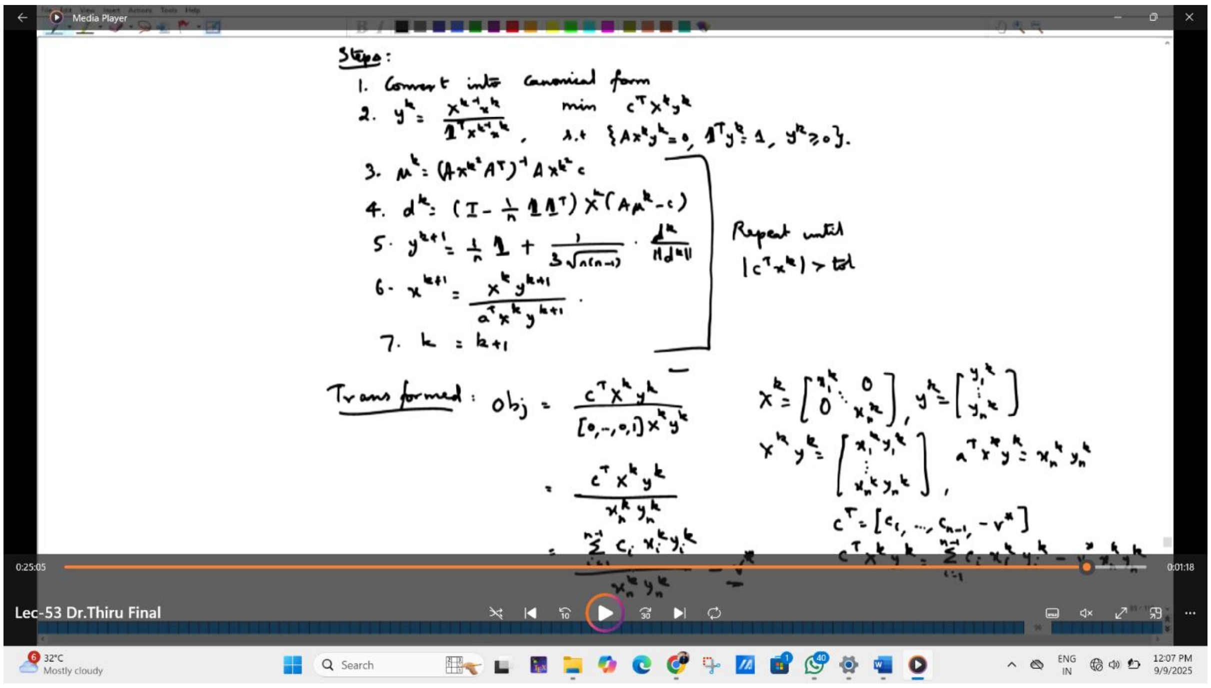
The c that you have here, c^T is actually c_1 ... c_{n-1} and finally you have a -v*.

The -v* was corresponding to the optimal value of the standard form.

So $c^T X_k y_k$, since $X_k y_k$ is as I have given here, you will just have it as $\Sigma c_i x_{i,k} y_{i,k}$ for i = 1 to n-1 minus $v^* x_{n,k} y_{n,k}$.

The objective function then becomes $(\Sigma c_i x_{i,k} y_{i,k}$ for i = 1 to n-1$) / (x_{n,k} y_{n,k}) - v^*$.

(Refer Slide Time 25:05)



Steps:

1. Convert into canonical form
2. $y^k = \frac{X^{k-1}x^k}{\mathbb{1}^T X^{k-1} x^k}$, $\quad \min \; c^T X^k y^k$ $\quad$ s.t $\{A X^k y^k = 0, \mathbb{1}^T y^k = 1, y^k \geq 0\}$.
3. $\mu^k = (A X^{k^2} A^T)^{-1} A X^{k^2} c$
4. $d^k = (I - \frac{1}{n}\mathbb{1}\mathbb{1}^T) X^k (A\mu^k - c)$
5. $y^{k+1} = \frac{1}{n}\mathbb{1} + \frac{1}{3\sqrt{n(n-1)}} \cdot \frac{d^k}{\|d^k\|}$
6. $x^{k+1} = \frac{X^k y^{k+1}}{a^T X^k y^{k+1}}$.
7. $k = k+1$

Repeat until $|c^T x^k| > tol$

Transformed: $\text{obj} = \frac{c^T X^k y^k}{[0,\ldots,0,1] X^k y^k}$

$= \frac{c^T X^k y^k}{x_n^k y_n^k}$

$= \frac{\sum_{i=1}^{n-1} c_i x_i^k y_i^k}{x_n^k y_n^k} - v^*$

$X^k = \begin{bmatrix} x_1^k & & 0 \\ & \ddots & \\ 0 & & x_n^k \end{bmatrix}$, $y^k = \begin{bmatrix} y_1^k \\ \vdots \\ y_n^k \end{bmatrix}$

$X^k y^k = \begin{bmatrix} x_1^k y_1^k \\ \vdots \\ x_n^k y_n^k \end{bmatrix}$, $\quad a^T X^k y^k = x_n^k y_n^k$

$c^T = [c_1, \ldots, c_{n-1}, -v^*]$

$c^T X^k y^k = \sum_{i=1}^{n-1} c_i x_i^k y_i^k - v^* x_n^k y_n^k$

0:25:05

0:01:18

Lec-53 Dr.Thiru Final

This occurs because of the special structure of a that we have chosen here. The a^T that we have chosen here is of the form all zeros and one.

Because of that, we can write the objective function in this particular form. If you are trying to minimize this, then of course this is actually a constant. This can be thrown away.

Minimizing this is equivalent to minimizing the numerator. That is the reason why we are actually sticking to this even when you do not know v*.

What I have told you is how do you get around the issue of using $c^T X_k y_k$ without knowing the value of v*.

We also need to get around the stopping criterion, which I will actually discuss in the next lecture. Thank you.

Note : $x_{n,k} = x_n^k$