Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 14

Hello everyone. This is the fourth lecture of third week. We were looking at optimization algorithms. We started with a very general framework of optimization algorithms where you start with an initial point, choose a descent direction and the step size and update the point based on the direction and the step size, you keep doing this iteratively until a stopping condition is met. For the previous two lectures we learnt different kinds of algorithms for computing the step size. Those are called line search algorithms. We learnt about exact line search algorithm and backtracking line search algorithm.

I would like to add a little more to exact line search algorithm before going to the next topic.

Recall that in exact line search algorithm, given x_k and d_k we define

$h(\alpha) = f(x_k + \alpha\, d_k)$

and we find that value of α for which $h(\alpha)$ is minimized.

One important question that we can ask is for which kind of functions can we actually use exact line search algorithm? We saw examples where we can use line search algorithms. For instance, if f is a function from R^2 to R and

$f(x_1, x_2) = x_1^2 + x_2^2$,

we saw that using exact line search algorithm was fine. This exact algorithm can be used, but if we had a more complicated function, then we saw that exact line search algorithm cannot be used because of intractability issues.

There is one class of functions where we can use exact line search algorithms without any issues: quadratic functions. The general form of a quadratic function when the domain is of dimension n is given by:

$f(x) = (1/2)\, x^T H x + b^T x + c$

Since we are actually trying to minimize f, the Hessian of f is actually H.

H needs to be a positive definite matrix, because in that case a unique minimizer exists. The values of b and c can be anything: b can be any n-dimensional vector and c can be any scalar.

Note: $x_k = x^k, d_k = d^k$

any quadratic function where H is positive definite, you can actually use exact line search and there will be no issues.

Let me derive the precise value of α for any function of this form. Given x_k and d_k, we have:

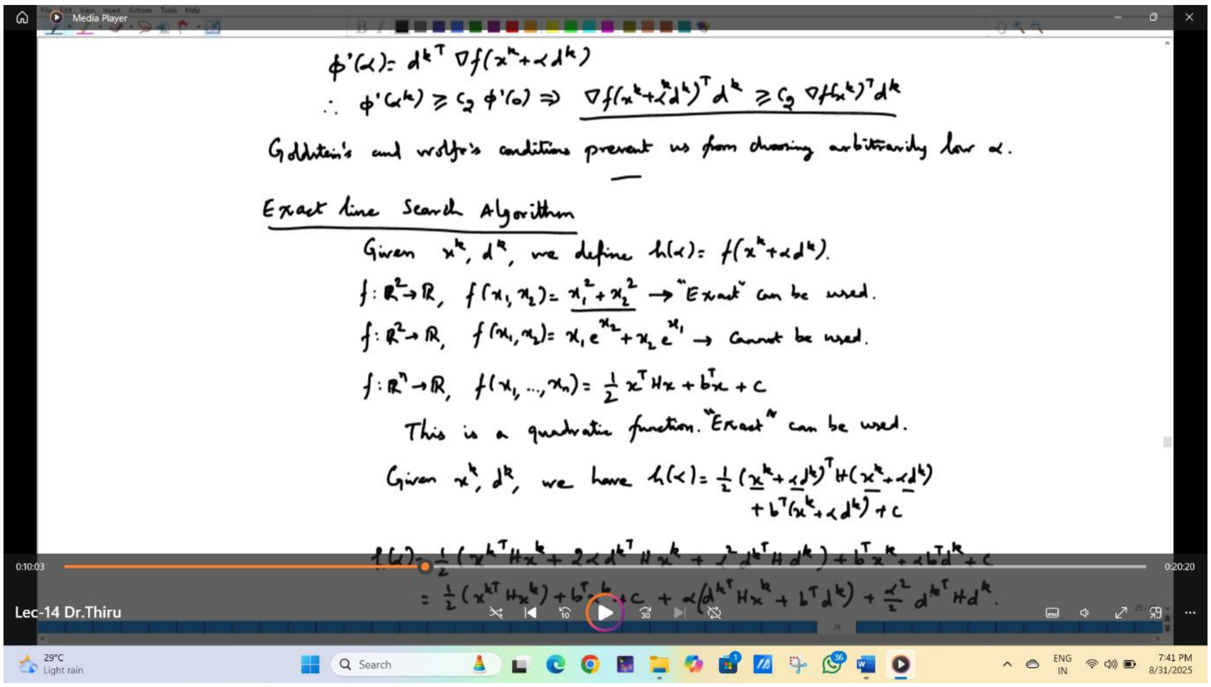
$h(\alpha) = f(x_k + \alpha\, d_k) = (1/2)(x_k + \alpha\, d_k)^T H\, (x_k + \alpha\, d_k) + b^T (x_k + \alpha\, d_k) + c$

Expanding this:

$h(\alpha) = (1/2)[x_k^T H\, x_k + 2\alpha\, d_k^T H\, x_k + \alpha^2\, d_k^T H\, d_k] + b^T x_k + \alpha\, b^T d_k + c$

$= (1/2)x_k^T H\, x_k + b^T x_k + c + \alpha\, d_k^T H\, x_k + (\alpha^2/2)\, d_k^T H\, d_k + \alpha\, b^T d_k$

(Refer Slide Time 10:03)



$\phi'(\alpha) = d^{k^T} \nabla f(x^k + \alpha d^k)$

$\therefore\ \phi'(\alpha^k) \geq c_2\, \phi'(0) \Rightarrow \underline{\nabla f(x^k + \alpha^k d^k)^T d^k \geq c_2\, \nabla f(x^k)^T d^k}$

Goldstein's and wolfe's conditions prevent us from choosing arbitrarily low α.

<u>Exact line Search Algorithm</u>

Given x^k, d^k, we define $h(\alpha) = f(x^k + \alpha d^k)$.

$f: \mathbb{R}^2 \to \mathbb{R},\ f(x_1, x_2) = \underline{x_1^2 + x_2^2} \to$ "Exact" can be used.

$f: \mathbb{R}^2 \to \mathbb{R},\ f(x_1, x_2) = x_1 e^{x_2} + x_2 e^{x_1} \to$ Cannot be used.

$f: \mathbb{R}^n \to \mathbb{R},\ f(x_1, \dots, x_n) = \frac{1}{2} x^T H x + b^T x + c$

This is a quadratic function. "Exact" can be used.

Given x^k, d^k, we have $h(\alpha) = \frac{1}{2}(\underline{x}^k + \underline{\alpha d}^k)^T H (\underline{x}^k + \alpha \underline{d}^k)$
$+ b^T(x^k + \alpha d^k) + c$

$h(\alpha) = \frac{1}{2}(x^{k^T} H x^k + 2\alpha d^{k^T} H x^k + \alpha^2 d^{k^T} H d^k) + b^T x^k + \alpha b^T d^k + c$

$= \frac{1}{2}(x^{k^T} H x^k) + b^T x^k + c + \alpha(d^{k^T} H x^k + b^T d^k) + \frac{\alpha^2}{2} d^{k^T} H d^k.$

Note that $(1/2)x_k^T H\, x_k + b^T x_k + c = f(x_k)$

Also, the gradient of f at x_k is: $\nabla f(x_k) = H\, x_k + b$ (let's denote this as g_k)

So we can rewrite h(α) as:

$h(\alpha) = f(x_k) + \alpha\, d_k^T g_k + (\alpha^2/2)\, d_k^T H\, d_k$

Now, to find which α minimizes h(α), we set $h'(\alpha) = 0$:

$d_k^T g_k + \alpha\, d_k^T H\, d_k = 0$

So, $\alpha = -\,(d_k^T g_k) / (d_k^T H\, d_k)$

This tells us that when you have a quadratic function in n dimensions (where H is positive definite), we actually have a closed form solution for the step size.

The step size is just $-(d_k^T g_k) / (d_k^T H d_k)$. This is something that we will use throughout the course. Whenever we have a quadratic function, we will use exact line search with this formula.

I want to introduce the notation g_k for $\nabla f(x_k)$, which is commonly used in optimization literature. With this notation, α becomes $-(g_k^T d_k) / (d_k^T H d_k)$.

Now let's get back to the original algorithm. The general form of the algorithm is: initialize x_0 and tolerance, and for each iteration, choose a descent direction d_k, choose a step size α_k using line search algorithms, update $x_{k+1} = x_k + \alpha_k d_k$, and stop when $\|\nabla f(x_k)\| \leq$ tolerance.

We are now comfortable with choosing α_k using exact or backtracking line search. The step we are not fine with yet is choosing the descent direction d_k.

The problem in choosing descent direction is that there are infinitely many possibilities. For example, with $f(x) = x_1^2 + x_2^2$ and $x_0 = (5,4)$, any direction d that satisfies $10d_1 + 8d_2 < 0$ is a descent direction. In general, for any f and x_k, the descent directions satisfy $\nabla f(x_k)^T d_k < 0$ (or $g_k^T d_k < 0$). This represents a half-plane in 2D, a half-space in higher dimensions - always an infinite set.

We need a specific way to choose a direction. We will look at two algorithms: steepest descent (or gradient descent) and conjugate gradient. We'll start with gradient descent.

The gradient descent algorithm chooses $d_k = -\nabla f(x_k) = -g_k$. Let's check if this is a descent direction:

$d_k^T g_k = -g_k^T g_k = -\|g_k\|^2 < 0$ (unless $g_k = 0$, in which case we've reached the optimum)

So $-g_k$ is always a descent direction.

Why is it the best? We can use the Cauchy-Schwarz inequality, which states that for any two vectors a and b:

$|a^T b| \leq \|a\| \|b\|$

With equality if and only if $a = \lambda b$ for some scalar λ.

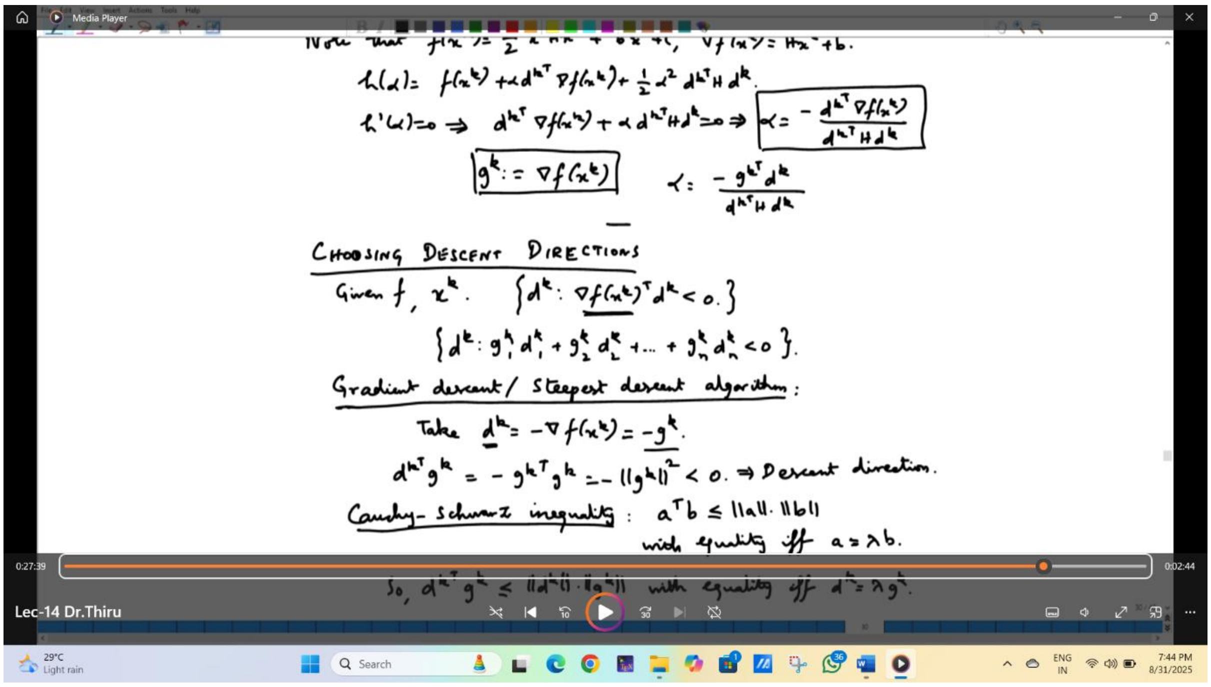
If we restrict ourselves to directions with fixed norm (say $\|d_k\| = 1$), then the descent rate is given by $d_k^T g_k$.

By Cauchy-Schwarz, $|d_k^T g_k| \leq \|d_k\| \|g_k\| = \|g_k\|$.

The most negative value (steepest descent) occurs when $d_k^T g_k = -\|g_k\|$, which happens when

$d_k = -g_k/\|g_k\|$.

(Refer Slide Time 27:39)


Note that $f(x) = \frac{1}{2} x^T H x + b^T x + c$, $\nabla f(x) = Hx + b$.

$$h(\alpha) = f(x^k) + \alpha d^{k^T} \nabla f(x^k) + \frac{1}{2} \alpha^2 d^{k^T} H d^k.$$

$$h'(\alpha) = 0 \Rightarrow d^{k^T} \nabla f(x^k) + \alpha d^{k^T} H d^k = 0 \Rightarrow \boxed{\alpha = -\frac{d^{k^T} \nabla f(x^k)}{d^{k^T} H d^k}}$$

$$\boxed{g^k := \nabla f(x^k)} \qquad \alpha = \frac{-g^{k^T} d^k}{d^{k^T} H d^k}$$

CHOOSING DESCENT DIRECTIONS

Given f, x^k. $\{d^k : \underline{\nabla f(x^k)}^T d^k < 0.\}$

$$\{d^k : g_1^k d_1^k + g_2^k d_2^k + \dots + g_n^k d_n^k < 0\}.$$

Gradient descent / Steepest descent algorithm:

Take $\underline{d}^k = -\nabla f(x^k) = \underline{-g^k}$.

$d^{k^T} g^k = -g^{k^T} g^k = -\|g^k\|^2 < 0. \Rightarrow$ Descent direction.

Cauchy-Schwarz inequality: $a^T b \leq \|a\| \cdot \|b\|$ with equality iff $a = \lambda b$.

So, $d^{k^T} g^k \leq \|d^k\| \cdot \|g^k\|$ with equality iff $d^k = \lambda g^k$.

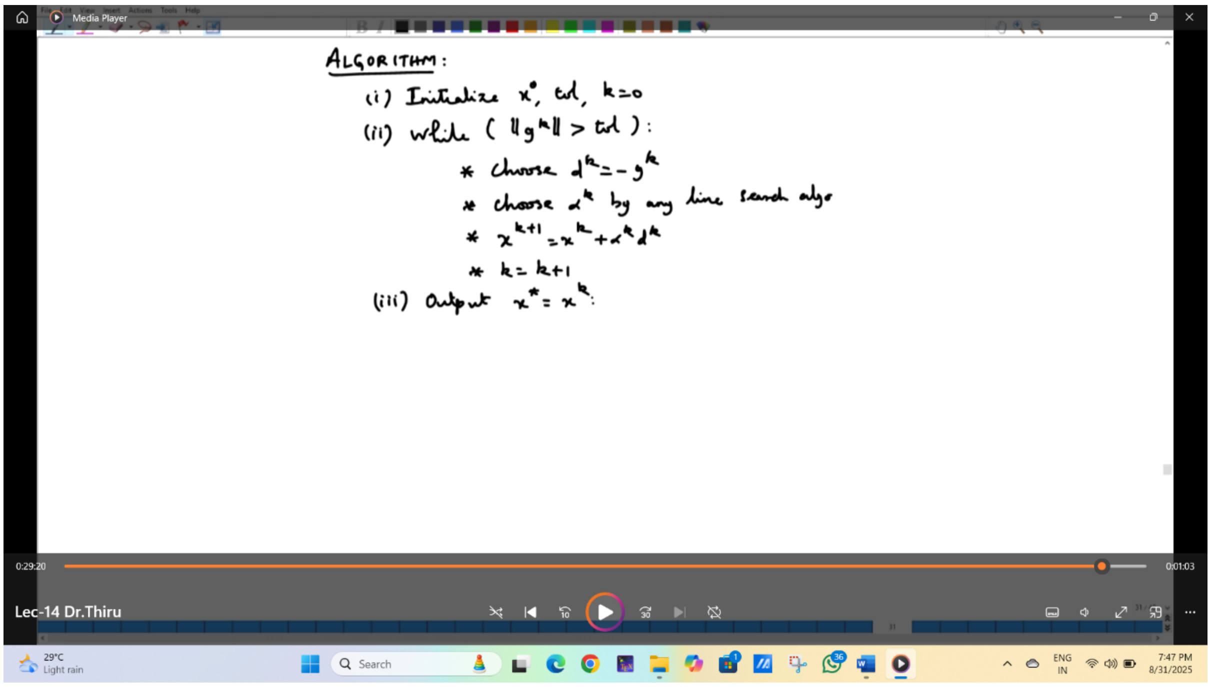
So choosing $d_k = -g_k$ gives the steepest possible descent, hence the name "steepest descent algorithm."

It's also called "gradient descent" because the direction is exactly the negative of the gradient.

The algorithm is as follows:

```
1. Initialize $x_0$, tolerance, and set $k = 0$
2. While $\|g_k\|$ > tolerance:
   a. $d_k = -g_k$
   b. Choose $\alpha_k$ using exact or backtracking line search
   c. $x_{k+1} = x_k + \alpha_k d_k = x_k - \alpha_k g_k$
   d. $k = k + 1$
3. Output $x^* = x_k$
```

(Refer Slide Time 29:20)



```
ALGORITHM:
(i) Initialize $x^0$, tol, $k=0$
(ii) While ( $\|g^k\| >$ tol ):
        * choose $d^k = -g^k$
        * choose $\alpha^k$ by any line search algo
        * $x^{k+1} = x^k + \alpha^k d^k$
        * $k = k+1$
(iii) Output $x^* = x^k$.
```

We will look at some examples of gradient descent algorithm in the next lecture. Thank you.