CS427: Mathematics for Data Science *Instructor: Prabuchandran K. J.*

Lecture on 06 April 2023

Lecture 32

Scribes: B. Siddharth Prabhu (200010003), Devesh Kumar (200030017)

1 Recap

In the previous lecture , we looked at line search, where each iteration computes a search direction d, and then finds an acceptable step size α, finally computing $x_{k+1} = x_k + \alpha_k d_k$, until some termination condition is met, like $\|\nabla f(x_k)\| < \epsilon$, which considers relative change, as below:

$$\left| \frac{f(x_{k+1}) + f(x_k)}{f(x_k)} \right| < \epsilon$$

We saw how exact search deals with finding $argmin_\alpha \Phi(\alpha)$ to calculate step size α_k. Then, we studied the characterization of descent direction, and the Armigo-Goldstein Conditions for Inexact Line Search.
In this Lecture, we shall cover the following topics:

- Line Search and Armijo Condition: Recap
- Backtracking Line Search
- Armijo-Wolfe Conditions
- The Zig-Zag Theorem
- Condition Number: An Intuitive View
- Cholesky Decomposition for Transformation
- A Note on Newton's Method
- Heavy Ball Momentum Method

(Note that the ordering of certain topics has been altered for cohesion and consistency purposes, in line with various reference materials.)

2 Line Search and Armijo Condition: Recap

From the last lecture, we know that $\Phi(\alpha_k) = f(x_k + \alpha_k d_k)$. (Note that we use the notation $f(x_k) = f_k$ in some equations.) The sufficiency condition, also called the *Armijo* condition, is given by:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla {f_k}^T d_k \quad = l(\alpha), \text{ (say)} \tag{1}$$

Here, $c_1 \in (0, 1)$. Also, in terms of Φ,

$$\Phi(\alpha_k) \leq \Phi(0) + c_1 \alpha_k \Phi'(0) \tag{2}$$

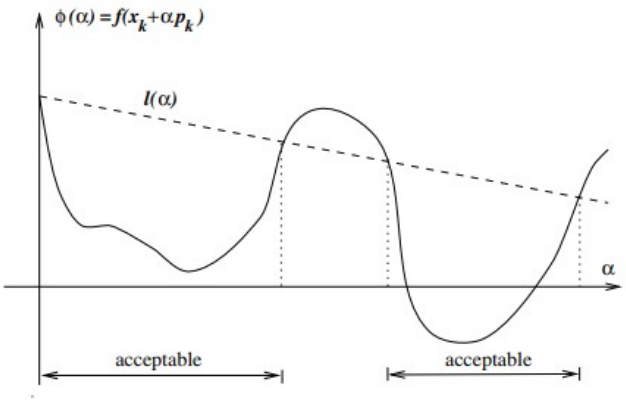


Figure 1: Sufficient Decrease (Armijo) condition

The Armijo condition from Equation (1) is illustrated in Figure (1). This condition states that α is acceptable only if $\Phi(\alpha) \leq l(\alpha)$. In other words, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla {f_k}^T d_k$. The function $l(\alpha)$ has negative slope, but lies above Φ for small positive α. In practice, c_1 is chosen to be $\approx 10^{-4}$.

We also looked at the Goldstein Condition, which is used to prevent smaller values of α_k, and is given by:

$$f(x_k + \alpha_k d_k) \geq f(x_k) + c_2 \alpha_k \nabla {f_k}^T d_k \ , \ \ 0 < c_1 < c_2 < 1 \tag{3}$$

We have mentioned that the sufficient decrease condition in Equation (1) alone is not sufficient to ensure that the algorithm makes reasonable progress along the given search direction. However, if the line search algorithm chooses its candidate step lengths appropriately, by using a so-called *backtracking approach*, we can just the sufficient decrease condition to terminate the line search procedure.

3 Backtracking Line Search

In its most basic form, backtracking proceeds as follows:

Choose $\overline{\alpha} > 0, \rho \in (0,1), c \in (0,1)$; Set $\alpha \leftarrow \overline{\alpha}$;
repeat until $f(x_k + \alpha d_k) \leq f(x_k) + c\alpha \nabla f_k^T d_k$
$\alpha \leftarrow \rho\alpha$;
end(repeat)
Terminate with $\alpha_k = \alpha$

Here, initial step length $\overline{\alpha}$ is chosen to be 1 in Newton and Quasi-Newton methods, but can have different values in other algorithms such as steepest descent or conjugate gradient. An acceptable step length will be found after a finite number of trials, because α_k will eventually become small enough that the sufficient decrease condition holds.

Note that ρ is a contraction factor (which wasn't talked much about in class), and is often allowed to vary at each iteration of the line search. For example, it can be chosen by safeguarded interpolation, as we describe later. We need ensure only that at each iteration we have $\rho \in [\rho_{lo}, \rho_{hi}]$, for some fixed constants $0 < \rho_{lo} < \rho_{hi} < 1$.

This simple and popularly used strategy for terminating a line search is well suited for Newton methods but is less appropriate for quasi-Newton and conjugate gradient methods.

4 Armijo-Wolfe Conditions

Recall that the Armijo condition given by: $f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla {f_k}^T d_k$, which ensures sufficient decrease. This sufficient decrease condition is not enough by itself to ensure that the algorithm makes reasonable progress because it is satisfied for all sufficiently small values of α. To rule out unacceptably short steps we introduce a second requirement, called the *curvature condition*, which is given by:

$$\nabla f_{k+1}^T d_k \geq c_2 \nabla f_k^T d_k \tag{4}$$

Here, $c_2 \in (c_1, 1)$. Note that the left-handside is simply the derivative $\Phi'(\alpha_k)$, so the curvature condition ensures that the slope of Φ at α_k is greater than c_2 times the initial slope $\Phi(0)$. This makes sense because if the slope $\Phi'(\alpha)$ is strongly negative, we have an indication that we can reduce f significantly by moving further along the chosen direction. On the other hand, if $\Phi'(\alpha_k)$ is only slightly negative or even positive, it is a sign that we cannot expect much more decrease in f in this direction, so it makes sense to terminate the line search. The curvature condition is illustrated in Figure (2), where p_k is used to denote d_k.

Note that typical values of c_2 are 0.9 when the search direction d_k is chosen by a Newton or quasi-Newton method, and 0.1 when d_k is obtained from a nonlinear conjugate gradient method.

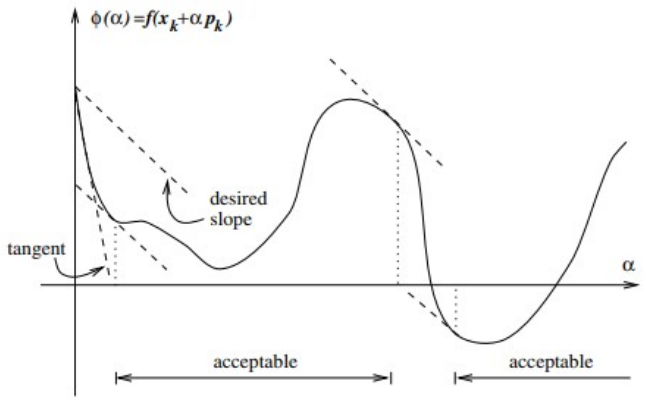


Figure 2: Curvature condition

The sufficient decrease and curvature conditions are known collectively as the *Wolfe conditions*. Let us collectively restate them, such that $0 < c_1 < c_2 < 1$:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla {f_k}^T d_k$$
$$\nabla f_{k+1}^T d_k \geq c_2 \nabla f_k^T d_k$$

A step length may satisfy the Wolfe conditions without being particularly close to a minimizer of Φ, as shown in Figure (3), where p_k is used to denote d_k.

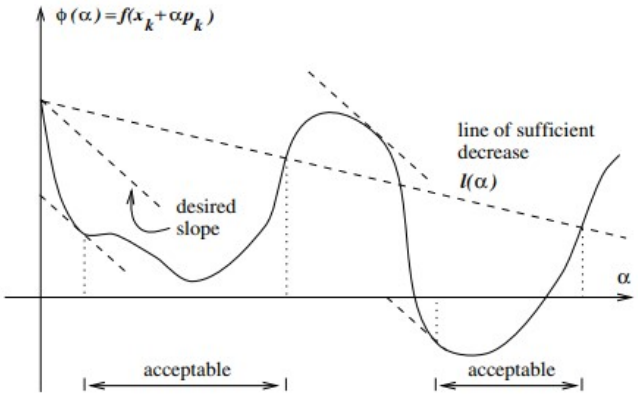


Figure 3: Satisfying values for Wolfe Condition

However, we can modify the curvature condition to force α_k to lie in at least a broad neighborhood of a local minimizer or stationary point of Φ. The modified condition set, given below, constitute the *strong Wolfe Conditions*.

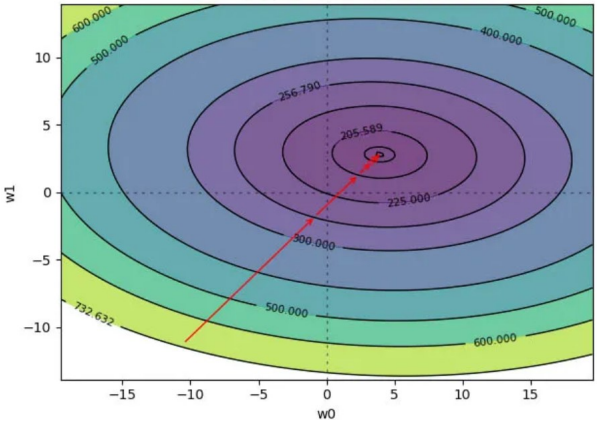
$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla {f_k}^T d_k$$
$$|\nabla f_{k+1}^T d_k| \leq c_2 |\nabla f_k^T d_k|$$

Here, $0 < c_1 < c_2 < 1$. Note that, in class, we had placed strict inequality in the curvature conditions, but the reference material suggests that ($\leq$) and modulus ($|\cdot|$) is sufficient for the strong Wolfe conditions. Hence, the strong Wolfe curvature condition restricts the slope of $\Phi(\alpha)$ from getting too positive, hence excluding points far away from the stationary point of Φ.

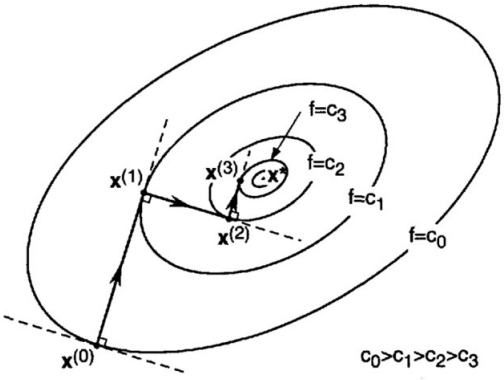
In most scenarios, Armijo conditions and Backtracking suffice for our purposes. Another thing to note: The Wolfe conditions are scale-invariant in a broad sense; Multiplying the objective function by a constant or making an affine change of variables does not alter them. They can be used in most line search methods, and are particularly important in the implementation of quasi-Newton methods.

5 The Zig-Zag Theorem

First, let us consider how steepest descent works with exact line search for quadratic functions, of the form $f(x) = x^T Ax + b^T x + c$. An iteration of gradient descent on this function would use the fact that $\nabla f(x) = 2Ax + b$, which would make the update rule $x_{k+1} = x_k - \alpha(2Ax_k + b)$.



(a) Circular Contours, Courtesy: Link



(b) Elliptical Contours, Courtesy: Link

Figure 4: Exact Line Search on different contours

Theorem 5.1. *Let $\{x_k\}$ be the sequence generated by the steepest descent algorithm. Then, for all k, $(x_{k+1} - x_k)$ is orthogonal to $(x_{k+2} - x_{k+1})$.*

Proof. If the contours are circular, then exact line search will reach the minimum in one iteration. So, let's consider elliptical contours. According to our iterations,

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$
$$x_{k+2} = x_{k+1} - \alpha_{k+1} \nabla f(x_{k+1})$$

Hence, let us find the inner product of the consecutive steps as,

$$\langle x_{k+1} - x_k,\ x_{k+2} - x_{k+1} \rangle \ = \ \langle \alpha_k \nabla f(x_k),\ \alpha_{k+1} \nabla f(x_{k+1}) \rangle$$

Further, the dot product is:

$$\langle x_{k+1} - x_k,\ x_{k+2} - x_{k+1} \rangle \ = \ \alpha_k \alpha_{k+1} \langle \nabla f(x_k),\ \nabla f(x_{k+1}) \rangle \tag{5}$$

Note that α_k minimizes $\Phi_k(\alpha) = f(x_k - \alpha f(x_k))$. Thus,

$$\frac{d\Phi_k}{d\alpha}(\alpha_k) = 0 \tag{6}$$

Though, using chain rule,

$$\frac{d\Phi_k}{d\alpha}(\alpha_k) = \langle -\nabla f(x_k),\ \nabla f(x_k - \alpha f(x_k)) \rangle$$

Hence,

$$\frac{d\Phi_k}{d\alpha}(\alpha_k) = \langle -\nabla f(x_k),\ \nabla f(x_k - \alpha f(x_k)) \rangle = \langle -\nabla f(x_k),\ \nabla f(x_{k+1}) \rangle \tag{7}$$

Thus, from Equations (6) and (7), we can conclude that the RHS of Equation (5) is zero. Hence, we arrive at our result:

$$\langle x_{k+1} - x_k,\ x_{k+2} - x_{k+1} \rangle \ = \ 0 \tag{8}$$

Thus, $(x_{k+1} - x_k)$ is orthogonal to $(x_{k+2} - x_{k+1})$. □

6 Condition Number: An Intuitive View

We define the condition number κ of a transformation matrix A using its highest and lowest valued eigenvalues, as:

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}}$$

For spherical contour-based transformations, the condition number would be smaller, at around 1. The larger the value of κ, the longer it would take for convergence using line search methods. This makes sense, since more ill-formed trough-like functions would take longer to reach the minima for. The graphic in Figure (5) shows such an intuition, although defining the transformation may not be a trivial task.

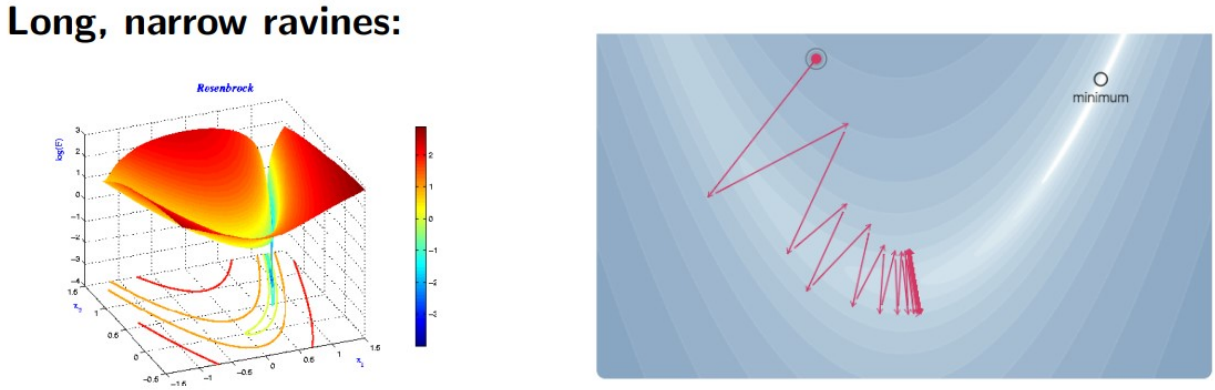


Figure 5: Ill-Conditioned Curvature

Thus, it would be better to transform elliptical contours to spherical contours before doing any line search. This is effectively what Newton's method does, as $x_{k+1} = x_k - \alpha_k H_f^{-1}(x_k)\nabla f(x_k)$.

7 Cholesky Decomposition for Transformation

So, how do we convert a function $f(x) = x^T Ax + b^T x + c$ to an equivalent $g(y) = y^T y + p^T y + q$? We can use the Cholesky decomposition, where we conduct a change of variables using $A = LL^T$ and $x = (L^T)^{-1}y$.

Our initial attempt in class went something along the lines of:

$$x^T Ax = y^T((L^T)^{-1})^T LL^T(L^T)^{-1}y = y^T((L^T)^{-1})^T Ly$$

Since $(L^T)^{-1} \neq (L^{-1})^T$ for all L, we can't reduce this directly to $y^T y$. Hence, the conversion should be something different if we want $y^T y$ to be generated exactly from $x^T Ax$.

However, the intuition is what's more important — we must convert an elliptical-contour based quadratic to a circular one before doing line search.

8 A Note on Newton's Method

We must remember that Newton's method converges quadratically only for points close to a local minimum. This makes sense since $H_f^{-1}(x)$ will be positive definite near the minimizer due to its nature of continuity. Also, at times, a correction factor is also added to force a decrease in function value, as shown in the update rule given by:

$$x_{t+1} = x_t - (H_f(x) + \alpha I)^{-1}\nabla f(x)$$

Convergence by this rule is provable, but outside the scope of this lecture.

9 Heavy Ball Momentum Method

This method of line search has a slightly different update rule that is impacted by its past velocity, accumulating past velocities as follows:

$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \beta(x_t - x_{t-1})$$

This would expand as:

$$\begin{aligned} x_{t+1} &= x_t - \alpha \nabla f(x_t) + \beta(x_t - x_{t-1}) \\ &= x_t - \alpha \nabla f(x_t) + \beta(-\alpha \nabla f(x_{t-1}) + \beta(x_{t-1} - x_{t-2})) \\ &= x_t - \alpha(\nabla f(x_t) + \beta \nabla f(x_{t-1}) + \beta^2 \nabla f(x_{t-2}) + ...) \end{aligned}$$

Consider an example, where $f(x) = x^2$. Here, $\nabla f(x) = 2x$. Then, the update rule is:

$$x_{t+1} = x_t - 2\alpha x_t + \beta(x_t - x_{t-1})$$

An image that demonstrates this is below, in Figure (6):

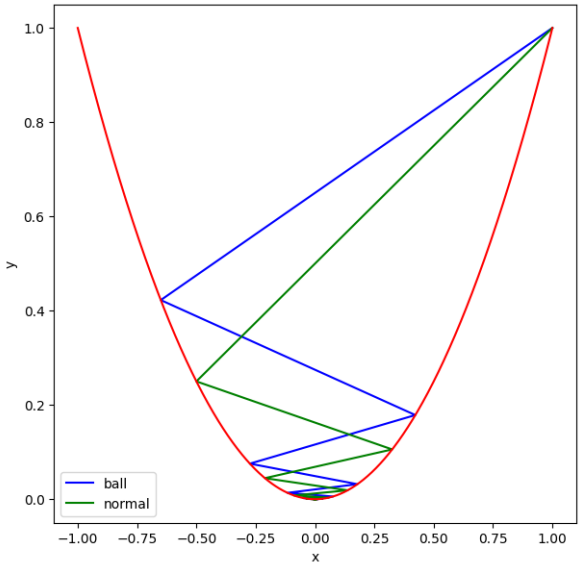


Figure 6: Heavy Ball Momentum Method on $f(x) = x^2$

A generalization of this would work out to be:

$$\begin{aligned} x_{t+1} &= x_t - \alpha m_{t+1} \\ m_{t+1} &= \beta m_{t-1} - \nabla f(x_t) \end{aligned}$$

This is similar to how more sophisticated algorithms work, like RMSprop, Adagrad, and Adam algorithm. This takes into account the geometric average of past gradients, moving along a sort of normalized gradient.

A visual aid for the convergence of such algorithms is available here. Adam algorithm is observed to be a mix of Heavy Ball Momentum method and RMSprop method. (Its exact pseudocode and logic is beyond the scope of this course, but is readily available online.) The aforementioned link shows how Adam and RMSprop perform better compared to some of the other algorithms. To see the difference in performances of Adam and RMSprop, look at this gif.

In the next class, we'll begin our journey into Constrained Optimization, where we are not free to look for the minimum value of a function for all $x \in \mathbb{R}^n$, but are constrained by some inequalities.

10 References

1. Alabdullatef, Layan. "Complete Guide to Adam Optimization." *Towards Data Science (Medium)*, 2 September 2020, Link
2. Cao, Lihe et al. "Line search methods." *Cornell University Computational Optimization Open Textbook*, 16 December 2021, Link
3. Nocedal, Jorge et al. "Numerical optimization." (2006) New York, NY: Springer. ISBN: 978-0-387-30303-1
4. Lecture Slides by Prof. Iman Shames for the course 'Introduction to Optimization' taught in 2019 (ELEN90026) at the University of Melbourne. Link
5. Lecture Slides by Prof. Jimmy Ba and Prof. Roger Grosse for the course 'Neural Networks and Deep Learning' taught in Winter 2019 (CSC 421/2516) at the University of Toronto. Link
6. Lecture Handouts by Prof. Amir Ali Ahmadi for the course 'Computing and Optimization' taught in Fall 2021 (ORF 363/COS 323) at Princeton University, NJ. Link

11 Related Code

Some of the plots in this lecture have been coded by us. Refer to this repo on GitHub!