3 Gradient Descent Method

- $\mathbf{p} = -\nabla f(\mathbf{x})$ Steepest descent direction.

If we take the direction that takes the steepest descent of f in the immediate neighborhood of $\mathbf{x}$ until we stop going descent directions, we are guaranteed to reach a local minima.

If we apply steepest descent to a quadratic function, then after many steps the algorithm takes alternate steps approximating two directions: those corresponding to the eigenvectors of the smallest and the largest eigenvalues of the Hessian matrix. The convergence rate can be shown to be linear:

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) \leq (\frac{\kappa - 1}{\kappa + 1})^2 (f(\mathbf{x}^k) - f(\mathbf{x}^*))$$

where κ is the ratio of the largest to the smallest eigenvalue of Hessian matrix $\mathbf{H}$. Considering $f(\mathbf{x}^k) - f(\mathbf{x}^*)$ as how accurate the solution is at iterate k. At each iteration, this is multiplied by a number less than 1. Therefore it will eventually go to 0. In general, if steepest descent is applied for strictly convex functions using a good line search, the convergence is linear.

Example: Implement gradient descent method with backtracking linesearch, where $c = 0.1, \rho = \frac{1}{2}$. Test it on the function

- $f(\mathbf{x}) = (x_1^2 + 10x_2^2)$, starting $\mathbf{x} = (50, 50)$.

- $f(\mathbf{x}) = e^{x_1 + 3x_2 - 0.1} + e^{x_1 - 3x_2 - 0.1} + e^{-x_1 - 0.1}$, starting $\mathbf{x} = (2.0, 1.0)$.

Conclusion: Advantages of gradient descent:

- Simple. No need to compute second-derivative (Hessian matrix). Computationally fast per iteration.
- Low storage: no matrices.

Disadvantages of gradient descent:

- Can be very, very slow.
- The direction is not well-scaled. Therefore the number of iterations largely depends on the scale of the problem.

Example: Test gradient descent method with backtracking linesearch, where $c = 0.1, \rho = \frac{1}{2}$ and $\alpha_0 = 1$ at each iteration. Test it on the function

- $f(\mathbf{x}) = 10(e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1})$, starting $\mathbf{x} = (2.0, 1.0)$.

How do we set the initial step-length?

Since gradient descent methods do not produce well-scaled search directions, it is important to use current information of the problem and the algorithm to make the initial guess.

- A popular strategy is to assume that the first-order change in the function at $\mathbf{x}^{(k)}$ will be the same as that obtained at the previous step. In other words, we choose α_0 so that $\alpha_0 \mathbf{p}^{(k)T}\nabla f(\mathbf{x}^{(k)}) = \alpha^{(k-1)}\mathbf{p}^{(k-1)T}\nabla f(\mathbf{x}^{(k-1)})$, so we have

$$\alpha_0 = \alpha^{(k-1)} \frac{\mathbf{p}^{(k-1)T}\nabla f(\mathbf{x}^{(k-1)})}{\mathbf{p}^{(k)T}\nabla f(\mathbf{x}^{(k)})}$$

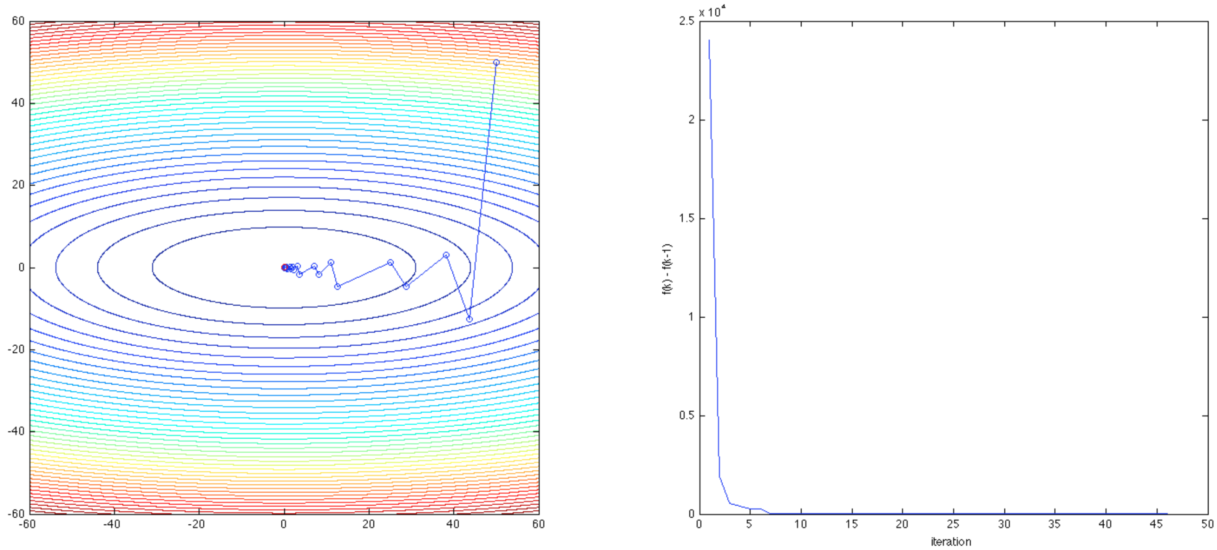
- Another useful strategy is to interpolate a quadratic to the data $f(\mathbf{x}^{(k-1)}), f(\mathbf{x}^{(k)})$ and $\phi'(0) = \mathbf{p}^{(k-1)T}\nabla f(\mathbf{x}^{(k-1)})$ and to define α_0 to be its minimizer:

$$\alpha_0 = \frac{2(f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k-1)}))}{\phi'(0)}$$

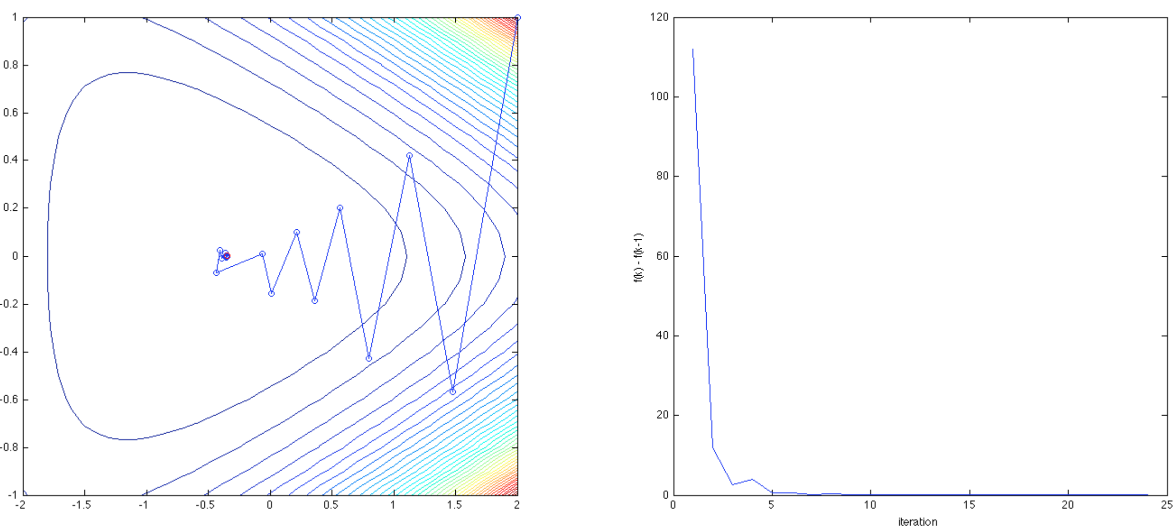
Example: Test gradient descent method with backtracking linesearch, where $c = 0.1, \rho = \frac{1}{2}$ and $\alpha_0 = \alpha^{(k-1)} \frac{\mathbf{p}^{(k-1)T}\nabla f(\mathbf{x}^{(k-1)})}{\mathbf{p}^{(k)T}\nabla f(\mathbf{x}^{(k)})}$ starting the 2nd iteration. Test it on the function

- $f(\mathbf{x}) = 10(e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1})$, starting $\mathbf{x} = (2.0, 1.0)$.

$*$ **Further Reading - Conjugate Gradient Method.**

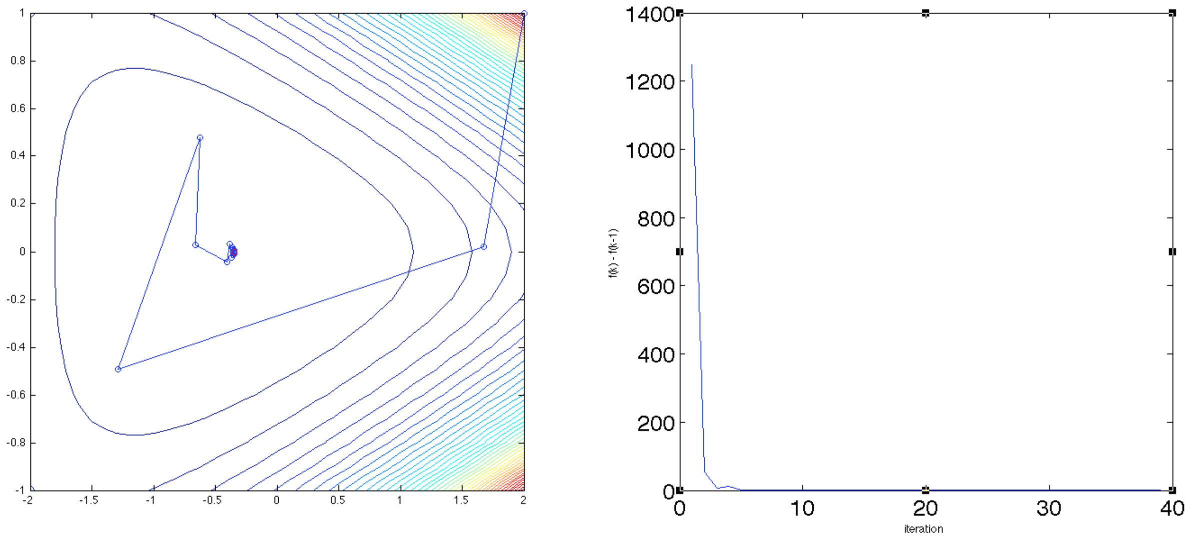


Gradient descent method on quadratic function

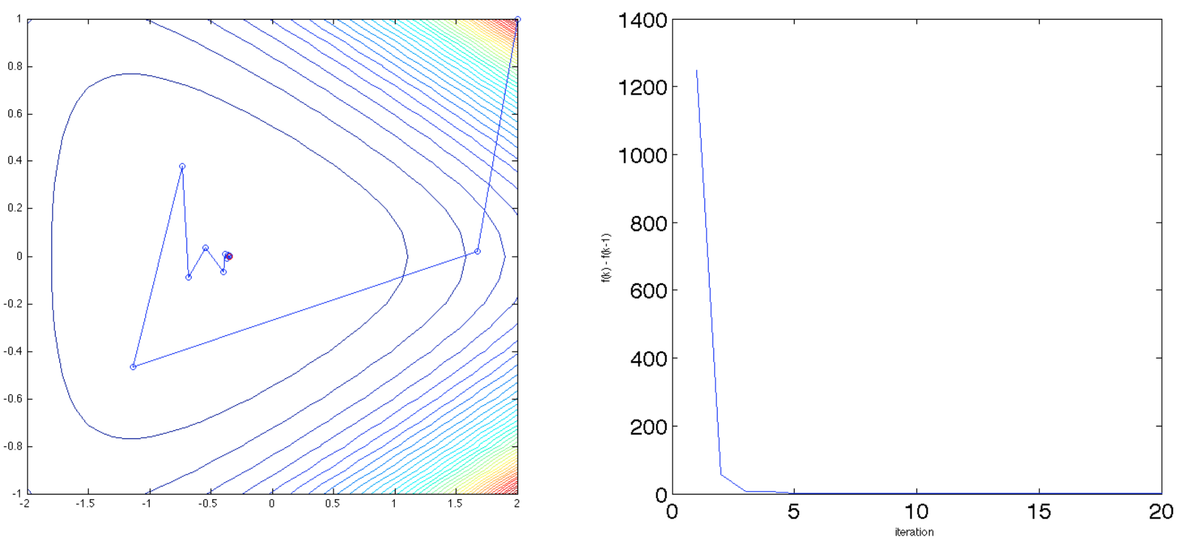


Gradient descent method on exponential function

Figure 3.1: Example of Gradient descent method performances.



Gradient descent with line search initialized with $\alpha = 1$ at each iteration



Gradient descent method with line search initialized with $\alpha_0 = \alpha^{(k-1)} \frac{\mathbf{p}^{(k-1)T} \nabla f(\mathbf{x}^{(k-1)})}{\mathbf{p}^{(k)T} \nabla f(\mathbf{x}^{(k)})}$

Figure 3.2: Example of Gradient descent method on 10-times scaled exponential function (example function 2).