Lecture 3 Unconstrained Optimization

Can Li

ChE 597: Computational Optimization
Purdue University

Unconstrained convex optimization

Consider unconstrained, smooth convex optimization

$$\min_x f(x)$$

That is, f is convex and differentiable with $\mathrm{dom}(f) = \mathbb{R}^n$. Denote optimal criterion value by $f^\star = \min_x f(x)$, and a solution by $x^\star$

Convex Functions: Local Minimum is Global Minimum

- For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, any local minimum is also a global minimum.

Proof: Let f be a convex function and x^* be a local minimum. By definition of convexity, for any two points $x, y \in \mathbb{R}^n$ and for any $\lambda \in [0, 1]$, we have:

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$$

Take $x = x^*$ and any $y \neq x^*$.

$$f(x^* + \lambda(y - x^*)) \leq (1-\lambda)f(x^*) + \lambda f(y)$$

since x^* is a local minimum, there exists $\lambda > 0$ such that $f(x^* + \lambda(y - x^*)) \geq f(x^*)$ Therefore we have

$$0 \leq f(x^* + \lambda(y - x^*)) - f(x^*) \leq \lambda(f(y) - f(x^*))$$

Given y is chosen arbitrarily, x must be the global optimum

Optimality condition

Theorem

For a differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$, a point x^* is a local (global) minimum if and only if $\nabla f(x^*) = 0$.

Proof.

Necessity: Assume x^* is a local minimum. If $\nabla f(x^*) \neq 0$, there exists some direction ($-\nabla f(x^*)$) where f decreases, contradicting x^* being a local minimum. Hence, $\nabla f(x^*) = 0$.
Sufficiency: Assume $\nabla f(x^*) = 0$. By the first-order condition of convexity,

$$f(x) \geq f(x^*) + \nabla f(x^*)^T (x - x^*),$$

which simplifies to $f(x) \geq f(x^*)$ since $\nabla f(x^*) = 0$. Therefore, x^* is a global (and thus local) minimum. □

Gradient Descent: An Overview

- Gradient Descent is an iterative optimization algorithm used to find the minimum of a function.
- **Basic Idea**: Move in the direction of the negative gradient of the function at the current point, because this is the direction of steepest descent.

Algorithm

Given a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, start from an initial point x_0 and iterate:

$$x_{k+1} = x_k - t_k \nabla f(x_k)$$

where t_k is the step size (learning rate) at iteration k.

- The choice of step size t_k is crucial for the convergence of the algorithm.
- Under appropriate conditions, gradient descent converges to a local minimum, which is global if f is convex.

Interpretation: Negative Gradient Direction as Steepest Descent

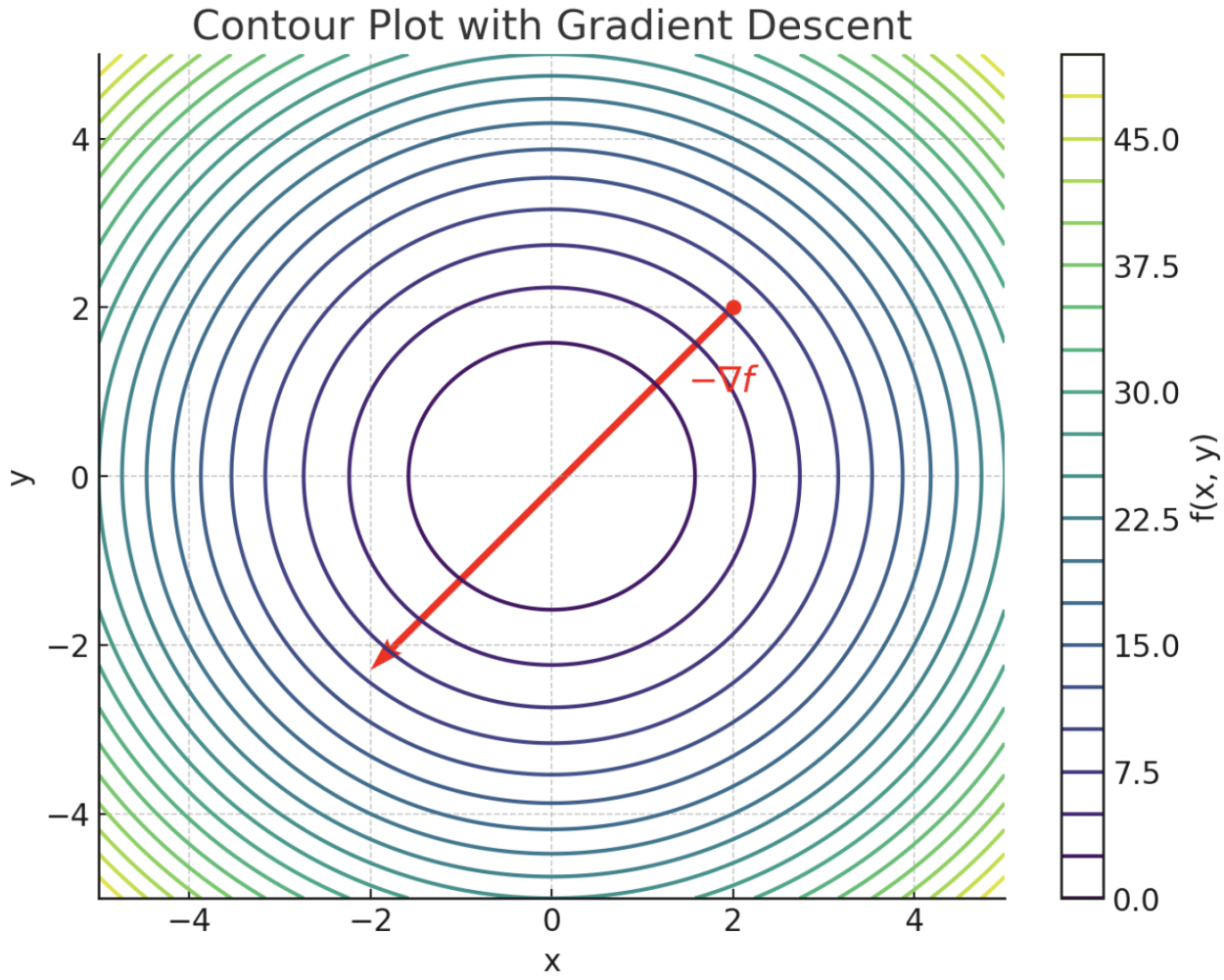
Proof:

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. The gradient of f at a point $\mathbf{x} \in \mathbb{R}^n$ is given by $\nabla f(\mathbf{x})$.

The directional derivative of f in the direction of a unit vector $\mathbf{u}$ is given by $D_{\mathbf{u}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{u}$.

To find the steepest descent, we need to minimize $D_{\mathbf{u}} f(\mathbf{x})$. This occurs when $\mathbf{u}$ is in the opposite direction to $\nabla f(\mathbf{x})$, proving that the gradient direction is the direction of steepest descent.

Visual Interpretation



Interpretation: Replace Hessian with Proximal terms

At each iteration, consider the expansion

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x)$$

Quadratic approximation, replacing usual Hessian $\nabla^2 f(x)$ by $\frac{1}{t} I$

$$f(x) + \nabla f(x)^T (y - x) \quad \text{linear approximation to } f$$

$$\frac{1}{2t}\|y - x\|^2 \quad \text{proximity term to } x, \text{ with weight } \frac{1}{2t}$$

Choose next point $y = x^+$ to minimize quadratic approximation:

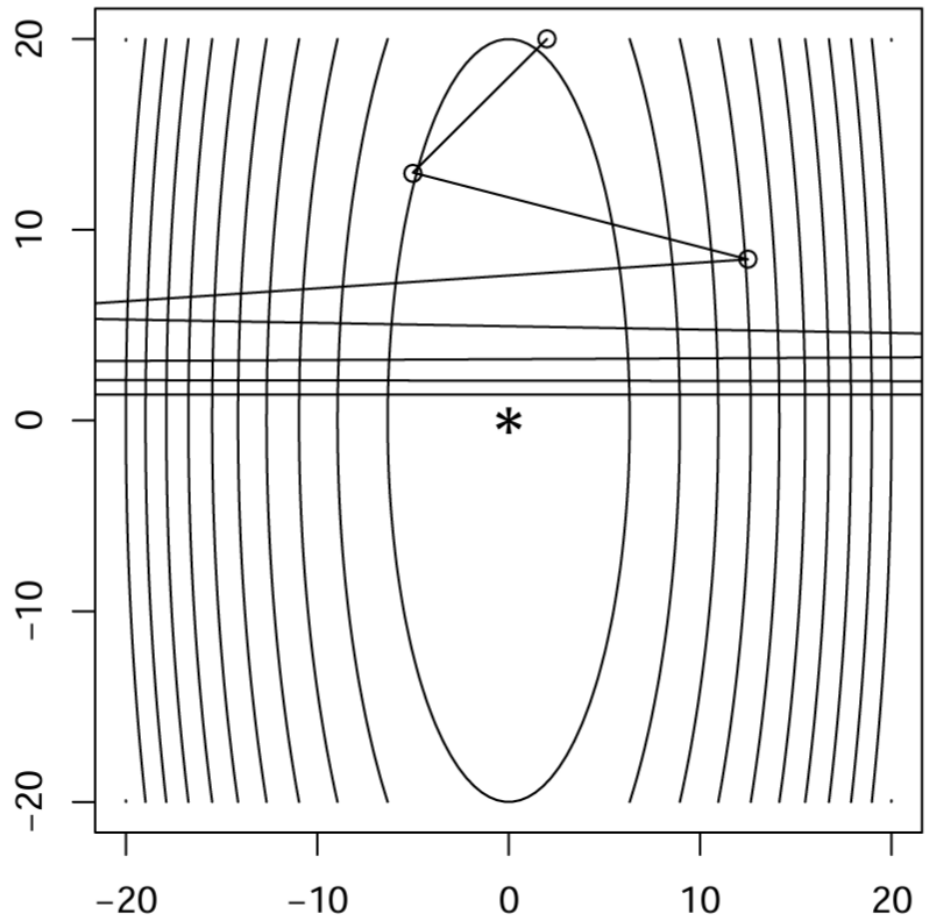
$$x^+ = x - t\nabla f(x)$$

How to choose step size t_k

Fixed step size

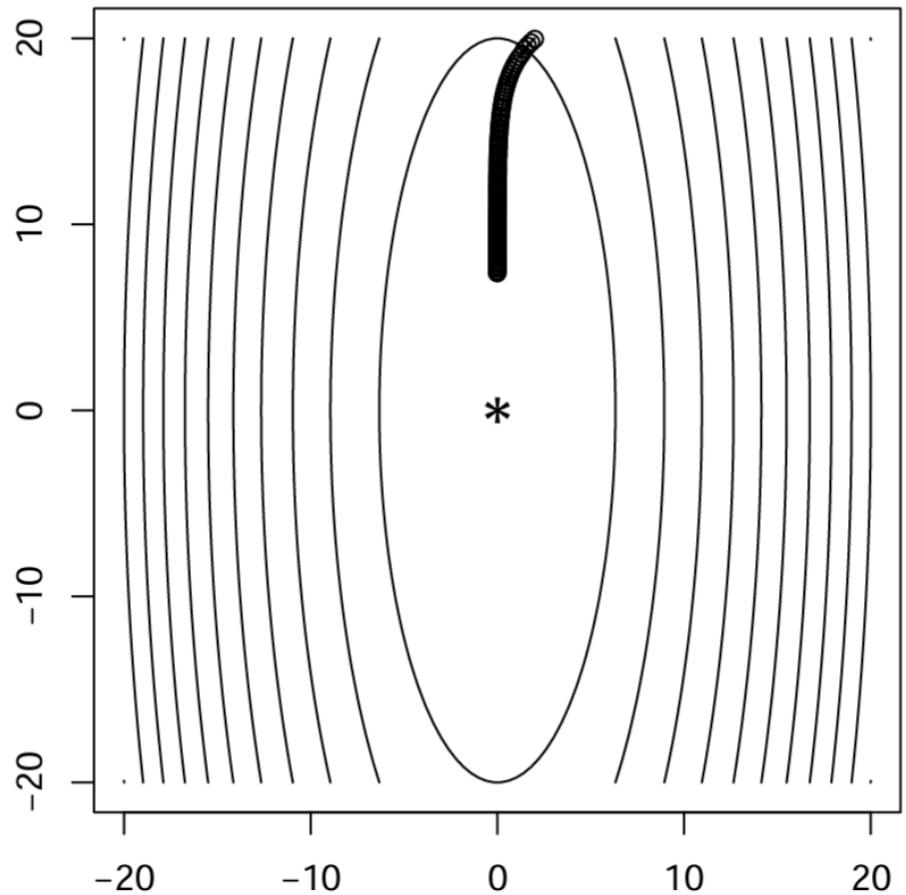
Simply take $t_k = t$ for all $k = 1, 2, 3, \ldots$, can diverge if t is too big.

Consider $f(x) = \left(10x_1^2 + x_2^2\right)/2$, gradient descent after 8 steps:



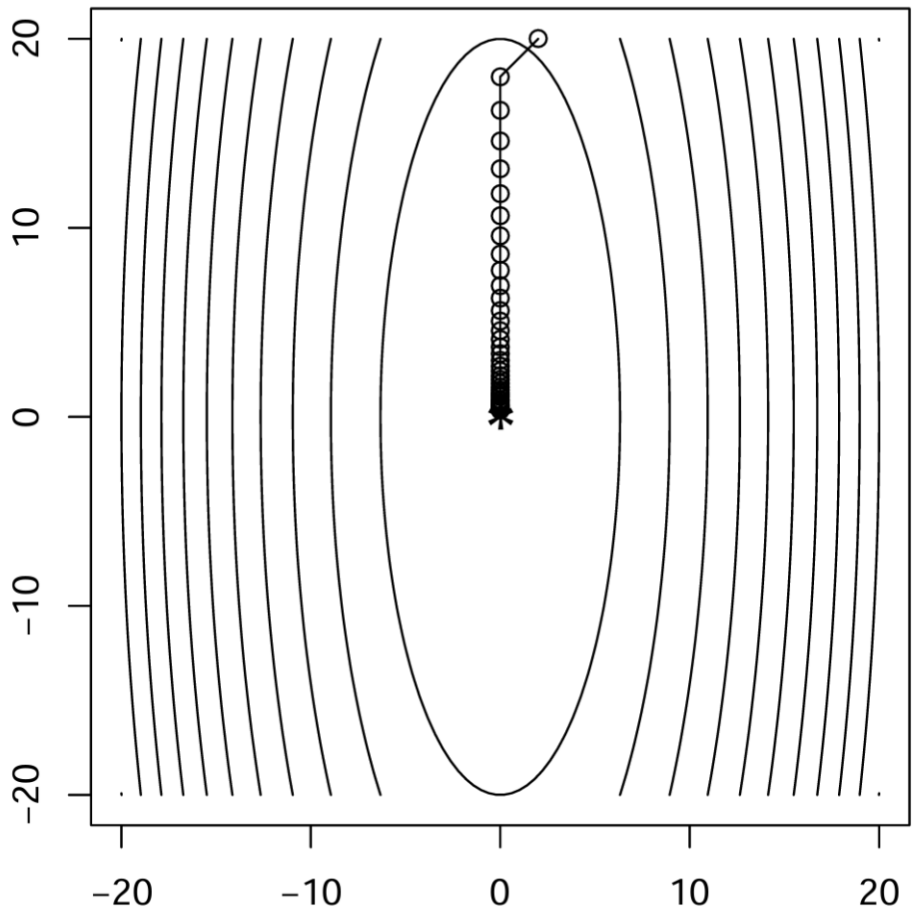
Fixed step size

Can be slow if t is too small. Same example, gradient descent after 100 steps:



Fixed step size

Converges nicely when t is "just right". Same example, 40 steps:



Backtracking Line Search

One way to adaptively choose the step size is to use **backtracking line search**:

- First fix parameters $0 < \beta < 1$ and $0 < \alpha \leq 1/2$
- At each iteration, start with $t = t_{\text{init}}$, and while

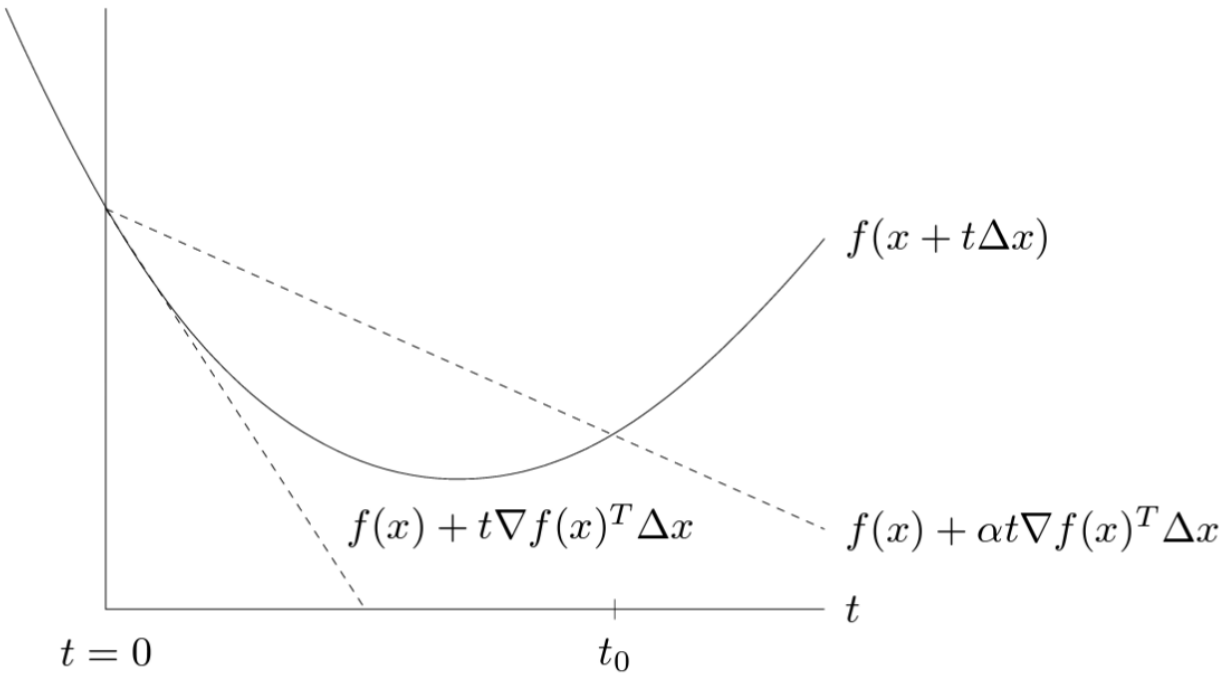
$$f(x - t\nabla f(x)) > f(x) - \alpha t \|\nabla f(x)\|^2$$

- shrink $t = \beta t$. Else perform gradient descent update

$$x^+ = x - t\nabla f(x)$$

Simple and tends to work well in practice (further simplification: just take $\alpha = 1/2$)

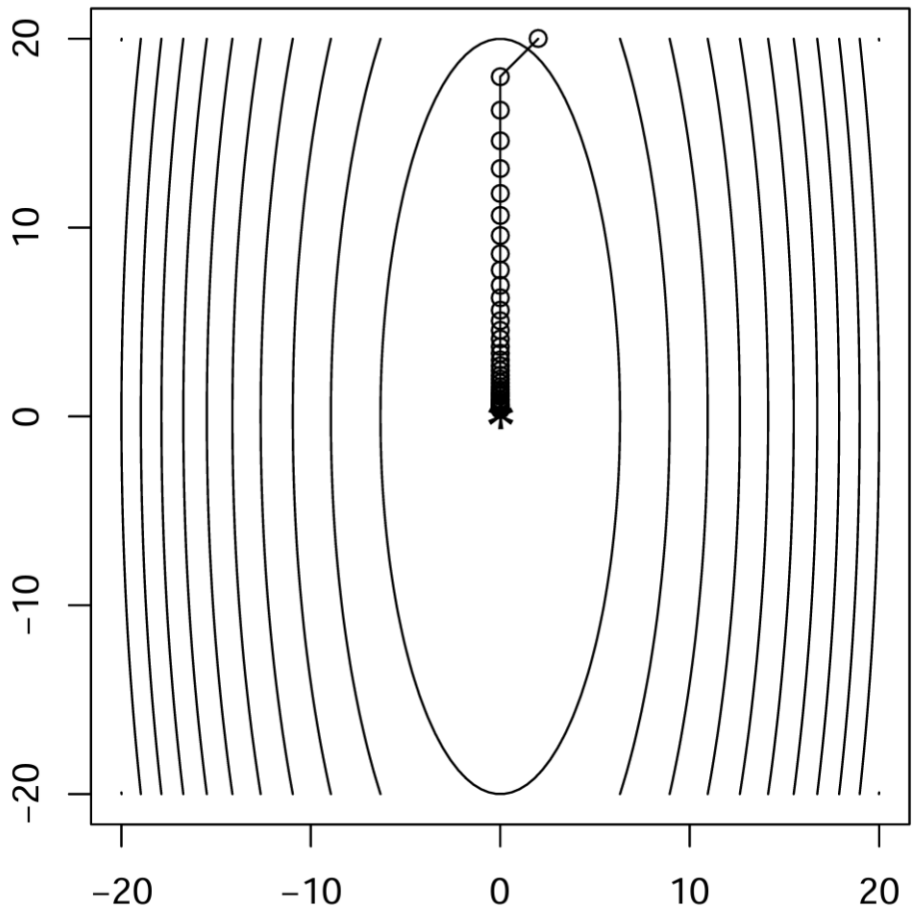
Backtracking interpretation



For us $\Delta x = -\nabla f(x)$

Backtracking example

Setting $\alpha = \beta = 0.5$, backtracking picks up roughly the right step size (12 outer steps, 40 steps total),



Exact line search

We could also choose the step to do the best we can along the direction of the negative gradient, called exact line search:

$$t = \underset{s \geq 0}{\operatorname{argmin}} f(x - s\nabla f(x))$$

Usually not possible to do this minimization exactly
Approximations to exact line search are typically not as efficient as backtracking, and it's typically not worth it

Convergence Analysis

Assume that f convex and differentiable, with $\mathrm{dom}(f) = \mathbb{R}^n$, and additionally that ∇f is Lipschitz continuous with constant $L > 0$, i.e.,
Lipschitz continuous definition:

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \quad \text{for any } x, y$$

(Or when twice differentiable: $\nabla^2 f(x) \leq LI$)

Theorem

Gradient descent with fixed step size $t \leq \frac{1}{L}$ satisfies

$$f(x^{(k)}) - f^* \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}$$

and the same result holds for backtracking, with t replaced by $\frac{\beta}{L}$
We say gradient descent has convergence rate $O(\frac{1}{k})$. That is, it finds ε-suboptimal point in $O(\frac{1}{\varepsilon})$ iterations

Newton's Method

Given unconstrained, smooth convex optimization:

$$\min_x f(x)$$

where f is convex, twice differentiable, and $\mathrm{dom}(f) = \mathbb{R}^n$. Recall that gradient descent chooses initial $x^{(0)} \in \mathbb{R}^n$, and repeats:

$$x^{(k)} = x^{(k-1)} - t_k \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

In comparison, **Newton's method** repeats:

$$x^{(k)} = x^{(k-1)} - (\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Here $\nabla^2 f(x^{(k-1)})$ is the Hessian matrix of f at $x^{(k-1)}$
Gradient descent is a **first-order method**. Newton's method is a **second-order method**

Interpretation: Second order approximation

Recall the motivation for gradient descent step at x: we minimize the quadratic approximation

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t}\|y - x\|^2$$

over y, and this yields the update $x^+ = x - t\nabla f(x)$.
Newton's method uses in a sense a **better quadratic approximation**

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x)$$

and minimizes over y to yield $x = x - (\nabla^2 f(x))^{-1} \nabla f(x)$.

Alternative Interpretation: Linearized Optimality Condition

Alternative interpretation of Newton step at x: we seek a direction v so that $\nabla f(x+v) = 0$. Let $F(x) = \nabla f(x)$. Consider linearizing F around x, via first-order approximation:

$$0 = F(x+v) \approx F(x) + DF(x)v$$

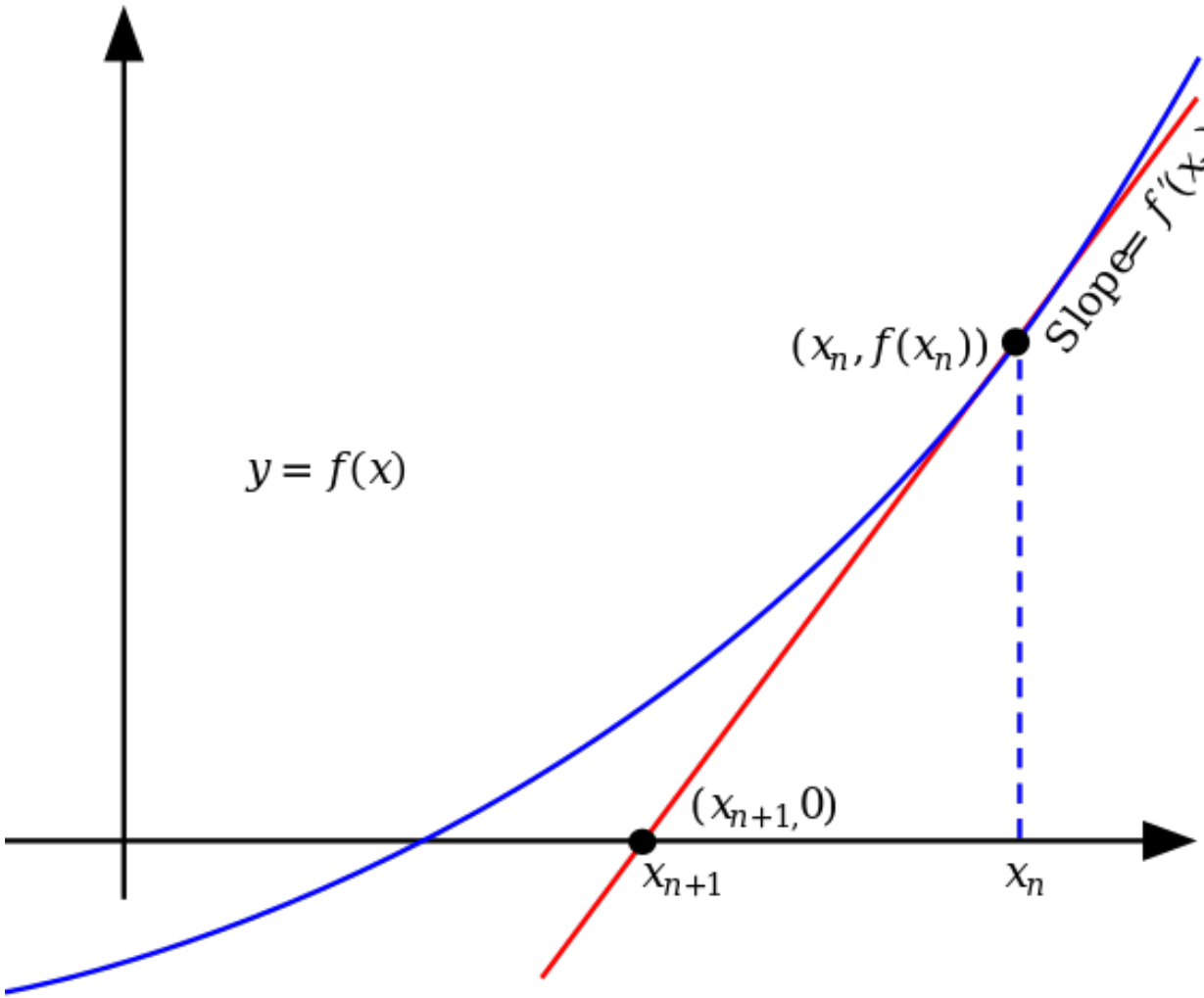
Solving for v yields $v = -(DF(x))^{-1}F(x) = -(\nabla^2 f(x))^{-1}\nabla f(x)$.

History

The work of Newton (1685) and Raphson (1690) originally focused on finding roots of polynomials. Simpson (1740) applied this idea to general nonlinear equations, and minimization by setting the gradient to zero.

Newton's method can be used to solve any nonlinear equations $F(x) = 0$

Geometric intuition



Affine Invariance of Newton's Method

Important property of Newton's method: **affine invariance**. Given f, nonsingular $A \in \mathbb{R}^{n \times n}$. Let $x = Ay$, and $g(y) = f(Ay)$. Newton steps on g are

$$\begin{aligned} y^+ &= y - (\nabla^2 g(y))^{-1} \nabla g(y) \\ &= y - (A^T \nabla^2 f(Ay) A)^{-1} A^T \nabla f(Ay) \\ &= y - A^{-1} (\nabla^2 f(Ay))^{-1} \nabla f(Ay) \end{aligned}$$

Hence

$$Ay^+ = Ay - (\nabla^2 f(Ay))^{-1} \nabla f(Ay)$$

i.e.,

$$x^+ = x - (\nabla^2 f(x))^{-1} \nabla f(x)$$

So progress is independent of problem scaling. This is not true for gradient descent!

Newton Decrement

At a point x, we define the **Newton decrement** as

$$\lambda(x) = \left(\nabla f(x)^T (\nabla^2 f(x))^{-1} \nabla f(x)\right)^{1/2}$$

This relates to the difference between $f(x)$ and the minimum of its quadratic approximation:

$$\begin{aligned} f(x) - \min_y \left(f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x) \right) \\ = f(x) - \left(f(x) - \frac{1}{2} \nabla f(x)^T (\nabla^2 f(x))^{-1} \nabla f(x) \right) \\ = \frac{1}{2} \lambda(x)^2 \end{aligned}$$

Therefore can think of $\frac{1}{2}\lambda(x)^2$ as an approximate upper bound on the suboptimality gap $f(x) - f^*$.

Backtracking Line Search

So far we've seen **pure Newton's method**. This need not converge. In practice, we use **damped Newton's method** (typically just called Newton's method), which repeats

$$x^+ = x - t(\nabla^2 f(x))^{-1} \nabla f(x)$$

Note that the pure method uses $t = 1$.
Step sizes here are chosen by **backtracking search**, with parameters $0 < \alpha \leq 1/2$, $0 < \beta < 1$. At each iteration, start with $t = 1$, while

$$f(x + tv) > f(x) + \alpha t \nabla f(x)^T v$$

we shrink $t = \beta t$, else we perform the Newton update. Note that here $v = -(\nabla^2 f(x))^{-1} \nabla f(x)$, so $\nabla f(x)^T v = -\lambda^2(x)$.

Convergence Analysis

Assume that f is convex, twice differentiable, having $\mathrm{dom}(f) = \mathbb{R}^n$, and additionally

- ∇f is Lipschitz with parameter L
- f is strongly convex with parameter m
- $\nabla^2 f$ is Lipschitz with parameter M

Theorem

Newton's method with backtracking line search satisfies the following two-stage convergence bounds

$$f(x^{(k)}) - f^* \leq \begin{cases} (f(x^{(0)}) - f^*) - \gamma k & \text{if } k \leq k_0 \\ \frac{2m^3}{M^2} \left(\frac{1}{2}\right)^{2^{k-k_0+1}} & \text{if } k > k_0 \end{cases}$$

Here $\gamma = \alpha\beta^2 m^2/L^2$, $\eta = \min\{1, 3(1-2\alpha)\}m^2/M$, *and* k_0 *is the number of steps until* $\|\nabla f(x^{(k_0+1)})\|_2 < \eta$

Comparison to First-Order Methods

At a high-level:

- **Memory**: each iteration of Newton's method requires $O(n^2)$ storage ($n \times n$ Hessian); each gradient iteration requires $O(n)$ storage (n-dimensional gradient).
- **Computation**: each Newton iteration requires $O(n^3)$ flops (solving a dense $n \times n$ linear system); each gradient iteration requires $O(n)$ flops (scaling/adding n-dimensional vectors).
- **Backtracking**: backtracking line search has roughly the same cost, both use $O(n)$ flops per inner backtracking step.
- **Conditioning**: Newton's method is not affected by a problem's conditioning, but gradient descent can seriously degrade.

Quasi-Newton Methods

If the Hessian is too expensive (or singular), then a quasi-Newton method can be used to approximate $\nabla^2 f(x)$ with $H \succ 0$, and we update according to

$$x^+ = x - tH^{-1}\nabla f(x)$$

- Approximate Hessian H is recomputed at each step. Goal is to make H^{-1} cheap to apply (possibly, cheap storage too).
- Convergence is fast: superlinear, but not the same as Newton. Roughly n steps of quasi-Newton make the same progress as one Newton step.
- Very wide variety of quasi-Newton methods; the common theme is to "propagate" computation of H across iterations.
  - BFGS (Broyden-Fletcher-Goldfarb-Shanno) method
  - DFP (Davidon-Fletcher-Powell) method