# Gradient Descent vs Newton's Method

Jiahui Shui
benshui@smail.swufe.edu.cn

Southwestern University of Finance and Economics
School of Finance

April 28, 2021

# Description

Gradient descent is based on the observation that if the multi-variable function $f(\boldsymbol{x})$ is defined and differentiable in a neighborhood of a point $\boldsymbol{a}$, then $f(\boldsymbol{x})$ decreases fastest if one goes from $\boldsymbol{a}$ in the direction of the negative gradient of $f$ at $-\nabla f(\boldsymbol{a})$. It follows that if

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma \nabla f(\boldsymbol{x}_n)$$

for a $\gamma \in \mathbb{R}_+$, then we have $f(\boldsymbol{x}_{n+1}) \leq f(\boldsymbol{x}_n)$
The stopping criterion is often of the form $||\nabla f(\boldsymbol{x})|| \leq \varepsilon$, where $\varepsilon$ is small and positive.

# Line Search

Line search aims to find a $\gamma$ satisfies

$$\gamma = \text{argmin}_{s \geq 0} f(\boldsymbol{x} - s\nabla f(\boldsymbol{x}))$$

An inexact way to find it is called *Backtracking line search*. Choose $\alpha \in (0, 0.5), \beta \in (0, 1)$

$$\textbf{while} \quad f(\boldsymbol{x} - \gamma \nabla f((x)) > f(\boldsymbol{x}) - \alpha\gamma ||\nabla f(\boldsymbol{x})||^2, \quad \gamma = \beta\gamma$$

# Example 1

Consider *Rosenbrock* function

$$f(x, y) = (1 - x)^2 + (y - x^2)^2$$

Then $\nabla f$ is

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^T = (2x - 2 - 400x(y - x^2), 200(y - x^2))$$

This function has a norrow curved valley which contains the minimum. The bottom of the valley is very flat. Because of the curved flat valley the optimization is zigzagging slowly with small step sizes towards the minimum.

# Example 1

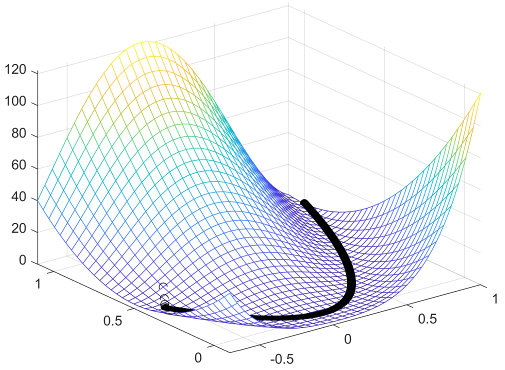



Figure 1: Gradient descent method for Rosenbrock function. Starting at $(-0.5, 0.5)$, $\gamma = 0.001$ and $\varepsilon = 0.0001$.
Result: $n = 20128, (x^*, y^*) = (0.999888, 0.999776)$, time $= 0.682328s$

# Example 1

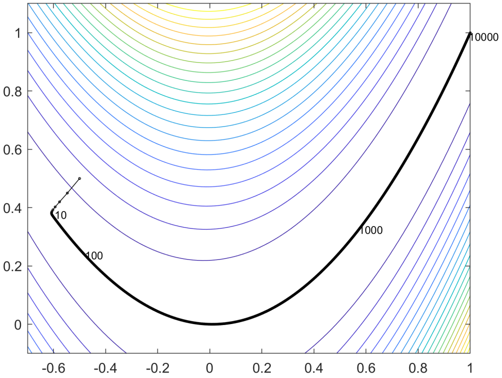



Figure 2: Contour line

# Description

Newton's method attempts to solve this problem by constructing a sequence $\{x_k\}$ from an initial guess (starting point) $x_0 \in \mathbb{R}$ that converges towards a minimizer $x^*$of $f$ by using a sequence of second-order Taylor approximations of $f$ around the iterates. The second-order Taylor expansion of $f$ around $x_k$ is

$$f'(x_k + t) = f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2 + o(t^2)$$

Its minimum can be found by setting the derivative to zero

$$f'(x_k) + f''(x_k)t = 0$$

Hence

$$t = -\frac{f'(x_k)}{f''(x_k)}$$

# Higher dimension

In higher dimension, the first order derivative changes into gradient, and the second order derivative changes into Hessian matrix. Then

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - [\nabla^2 f(\boldsymbol{x}_n)]^{-1} \nabla f(\boldsymbol{x}_n)$$

Often Newton's method is modified to include a small step size $0 < \gamma \leq 1$ instead of $\gamma = 1$.

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma [\nabla^2 f(\boldsymbol{x}_n)]^{-1} \nabla f(\boldsymbol{x}_n)$$

# Example 2

For a quadratic function

$$f(x, y) = ax^2 + bxy + cy^2$$

The Newton's method takes only one step and reaches the minimum. Since

$$\nabla f(\boldsymbol{x}) = (2ax + by, 2cy + bx)^T$$

and

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} 2a & b \\ b & 2c \end{pmatrix}$$

Then

$$[\nabla^2 f(\boldsymbol{x})]^{-1} \nabla f(\boldsymbol{x}) = (x, y)^T$$

# Newton's method for Example 1

Set $\gamma = 0.5$, the results are:

$$n = 42, \text{time} = 0.400881s$$

And

$$(x^*, y^*) = (0.9999980, 0.9999958)$$

# Newton's method for Example 1

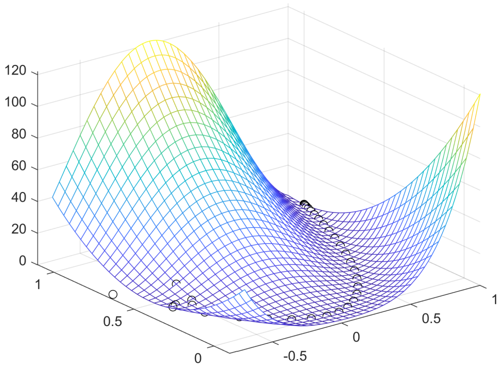



Figure 3: Newton's method for Rosenbrock function

# Newton's method for Example 1

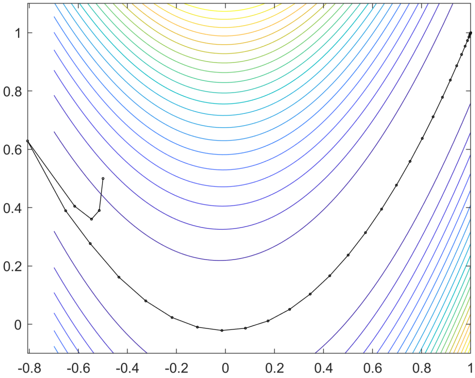



Figure 4: Contour line

# Comparison

| | Gradient descent | Newton's Method |
|---|---|---|
| Criterion | smooth $f$ | twice smooth $f$ |
| Iteration cost | cheap (compute gradient) | moderate to expensive |
| Rate | $O(1/\varepsilon)$ (acceleration: $O(1/\sqrt{\varepsilon})$ strong convexity: $O(\log(1/\varepsilon))$ | $O(\log\log(1/\varepsilon))$ |

# Comparison

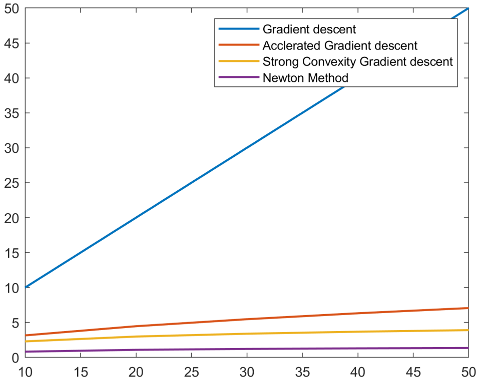



Figure 5: The rate of Gradient descent and Newton's method

# Example 3

Consider

$$f(x,y) = \frac{1}{2}(10x^2 + y^2) + 5\log(1 + e^{-x-y})$$

We have

$$\nabla f(x,y) = \left(10x - \frac{5e^{-x-y}}{1+e^{-x-y}},\, y - \frac{5e^{-x-y}}{1+e^{-x-y}}\right)^T$$

and

$$\nabla^2 f(x,y) = \begin{pmatrix} 10 + \frac{5e^{-x-y}}{(1+e^{-x-y})^2} & \frac{5e^{-x-y}}{(1+e^{-x-y})^2} \\ \frac{5e^{-x-y}}{(1+e^{-x-y})^2} & 1 + \frac{5e^{-x-y}}{(1+e^{-x-y})^2} \end{pmatrix}$$

Then $f$ is convex.

# Example 3

We start at $(x_0, y_0) = (20, 20)$. For gradient descent, we set $\gamma_{gd} = 0.05$. For Newton's method, we set $\gamma_{nt} = 0.5$.
And set the same $\varepsilon = 0.00001$(tolerance).
The result shows that gradient descent takes $177$ steps, $0.022490s$ but Newton's method takes $22$ steps and $0.012445s$ only.

# Example 3

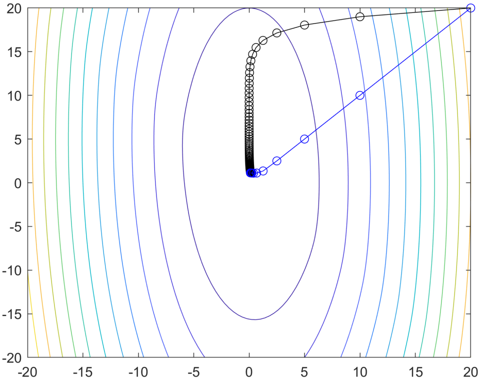



Figure 6: Comparison between Gradient descent and Newton's method

# Model

Estimate Logistic equation

$$p(\hat{y}) = \frac{1}{1 + e^{-\hat{y}}}$$

where $\hat{y}$ is given by

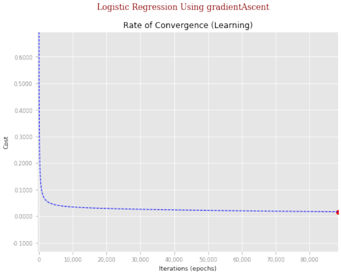
$$\hat{y} = \Theta^T X + \varepsilon = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

And estimates are trained using optimization of the conditional maximum Likelihood (cost) function

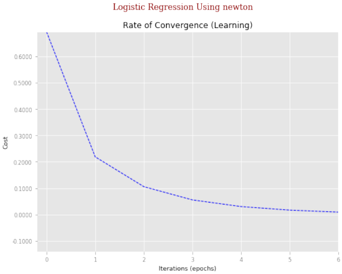
$$L(\Theta; y_n; x_n) = \prod_{n=1}^{N} [p(\hat{y}_n)]^{y_n} [1 - p(\hat{y}_n)]^{1-y_n}$$

$$\ell(\Theta; y_n; x_n) = -\sum_{n=1}^{N} (y_n \log[p(\hat{y_n})] + (1 - y_n) \log[1 - p(\hat{y}_n)])$$

# Result




(a) Gradient descent




(b) Newton's Method

Figure 7: Rate of Convergence

# Reference

[1] Boyd, Stephen, Stephen P. Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.
[2] https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization
[3]https://en.wikipedia.org/wiki/Gradient_descent
[4]https://github.com/DrIanGregory/MachineLearning-LogisticRegressionWithGradientDescentOrNewt