Newton's Algorithm

George Michailidis

gmichail@ucla.edu

STAT 102B

Brief Recap of Gradient Descent

Objective:

$$\min_x f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and f differentiable

1. Select $x_0 \in \mathbb{R}^n$
2. While stopping criterion>tolerance do:
   - (Optional) Select step size η_k adaptively
   - $x_{k+1} = x_k - \eta_k \nabla f(x_k)$
   - Calculate the value of the stopping criterion

Illustration of the GD Rationale

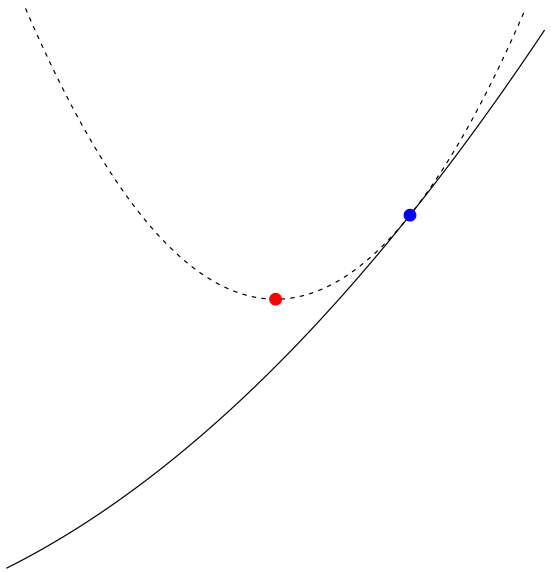

Figure 1: Blue point is x_k; Red point is x_{k+1}, the minimizer of the function $h(z) = f(x_k) + [\nabla f(x_k)]^\top (z - x_k) + \frac{1}{\eta}||z - x_k||_2^2$

The GD algorithm finds the minimum of a local quadratic approximation of the $f(x)$ function at the current update x_k; the approximation depends on the step size selected

Remarks

- Gradient descent uses a local quadratic approximation that only depends on the step size and does not take into consideration the local curvature of the function at the current iterate x_k
- Question: Can we modify the gradient descent algorithm to incorporate such information?

Newton's Algorithm - I

Consider that $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable everywhere in $\mathbb{R}^n$

Consider an initial point $x_0 \in \mathbb{R}^n$

Question: how shall we select the next iterate x_1?

Consider the second Taylor order expansion around x_0

$$f(x_1) \approx f(x_0) + \left[\nabla f(x_0)\right]^\top (x_1 - x_0) + \frac{1}{2}(x_1 - x_0)^\top \nabla^2 f(x_0)(x_1 - x_0),$$

and since we want $f(x_1) < f(x_0)$ we obtain that

$$\left[\nabla f(x_0)\right]^\top (x_1 - x_0) + \frac{1}{2}(x_1 - x_0)^\top \nabla^2 f(x_0)(x_1 - x_0) < 0$$

Newton's Algorithm - II

To find the optimal x_1 we take the derivative with respect to x_1 of

$$\frac{d}{dx_1}\left[f(x_0) + \left[\nabla f(x_0)\right]^\top (x_1 - x_0) + \frac{1}{2}(x_1 - x_0)^\top \left[\nabla^2 f(x_0)\right](x_1 - x_0)\right]$$

$$= \nabla f(x_0) + \nabla^2 f(x_0)(x_1 - x_0)$$

set it to zero and solve for x_1; namely,

$$\nabla f(x_0) + \left[\nabla^2 f(x_0)\right](x_1 - x_0) = 0 \Longrightarrow x_1 = x_0 - \left[\nabla^2 f(x_0)\right]^{-1} \nabla f(x_0)$$

Note that the inverse of the Hessian exists, since we assumed that the function f is everywhere twice differentiable in its domain

Newton's Algorithm - III

1. Select $x_0 \in \mathbb{R}^n$
2. While stopping criterion>tolerance do:
   - $x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$
   - Calculate the value of the stopping criterion

Newton's Algorithm - IV

Question:
Is $d_k = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$ a descent direction

Recall that for a direction d_k to be a descent direction, the following relationship needs to hold for all $x \in \mathbb{R}^n$:

$$\left[\nabla f(x_k)\right]^\top d_k < 0, \tag{1}$$

which for the proposed direction becomes

$$-\left(\left[\nabla f(x_k)\right]^\top \left[\nabla f^2(x_k)\right]^{-1} \nabla f(x_k)\right) < 0, \tag{2}$$

Note that the expression in parenthesis is a positive definite quadratic form for all $f(x_k) \neq 0$, since the Hessian is positive definite, and hence the result follows

Newton's Algorithm - V

Remarks:

- Newton's algorithm, originated with Isaac Newton in the 1660s. He initially developed it to find roots of polynomial equations, essentially solving for where a polynomial function equals zero
- It was later refined and simplified by Joseph Raphson in 1690, giving it its current form, and is also known as the Newton-Raphson method
- Thomas Simpson extended its use to solve general nonlinear equations in 1740, and recognized its applicability to optimization problems by finding where the gradient of a function becomes zero
- The version presented in slide 7 –known as the pure Newton's algorithm– does not use a step size, but works well primarily for quadratic functions
- The modern implementation of Newton's algorithm, called the guarded or damped Newton algorithm, uses a step size

Damped version of Newton's Algorithm

1. Select $x_0 \in \mathbb{R}^n$
2. While stopping criterion>tolerance do:
   - Select the step size η_k
   - $x_{k+1} = x_k - \eta_k \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$
   - Calculate the value of the stopping criterion

- The step size η_k can be selected using the backtracking line search strategy
- Another popular strategy is to use a schedule that decreases the value of η_k at each iteration; e.g., $\eta_k = \frac{\eta_0}{1+\lambda k}, \lambda > 0$

Newton's Algorithm for quadratic functions - I

Consider the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) = x^\top Q x + b^\top x + c, \tag{3}$$

with $Q \in \mathbb{R}^{n \times n}$ being a positive definite matrix, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$

We can derive the optimal $x^\star$ as follows:

$$\nabla f(x) = 0 \Longrightarrow Qx + b = 0 \Longrightarrow x^\star = -Q^{-1} b \tag{4}$$

Since Q is assumed to be positive definite, $x^\star$ is the unique global minimum

Newton's algorithm for quadratic functions - II

Consider an arbitrary starting point $x_0 \in \mathbb{R}^n$ and let us examine one iteration of Newton's algorithm

$$x_1 = x_0 - \left[\nabla f^2(x_0)\right]^{-1} \nabla f(x_0) \Longrightarrow x_1 = x_0 - Q^{-1}\left(Qx_0 + b\right) = -Q^{-1}b \qquad (5)$$

Hence, for quadratic functions, Newton's algorithm converges in a single iteration!

Illustration - I

Consider a data set comprising $n = 500$ observations and $p = 200$ predictors

There is some multicollinearity in the data;
one metric used to reflect that is the condition number, defined as

$$\text{CN} = \frac{\lambda_{\max}(\frac{X^\top X}{n})}{\lambda_{\min}(\frac{X^\top X}{n})}$$

For this data set, CI≈ 200

The results based on backtracking line search for GD and damped Newton are shown next

Illustration - II

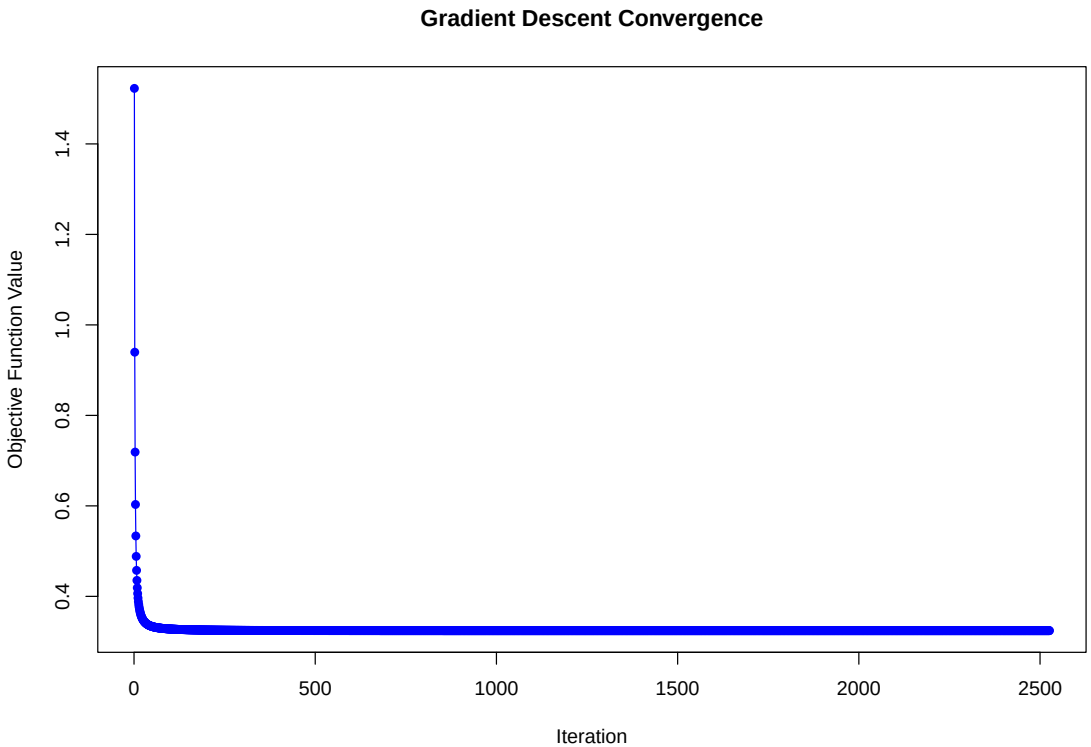

Figure 2: $\|\beta_{\text{true}} - \beta_{\text{GD}}\|_2 = 0.72$

Illustration - III

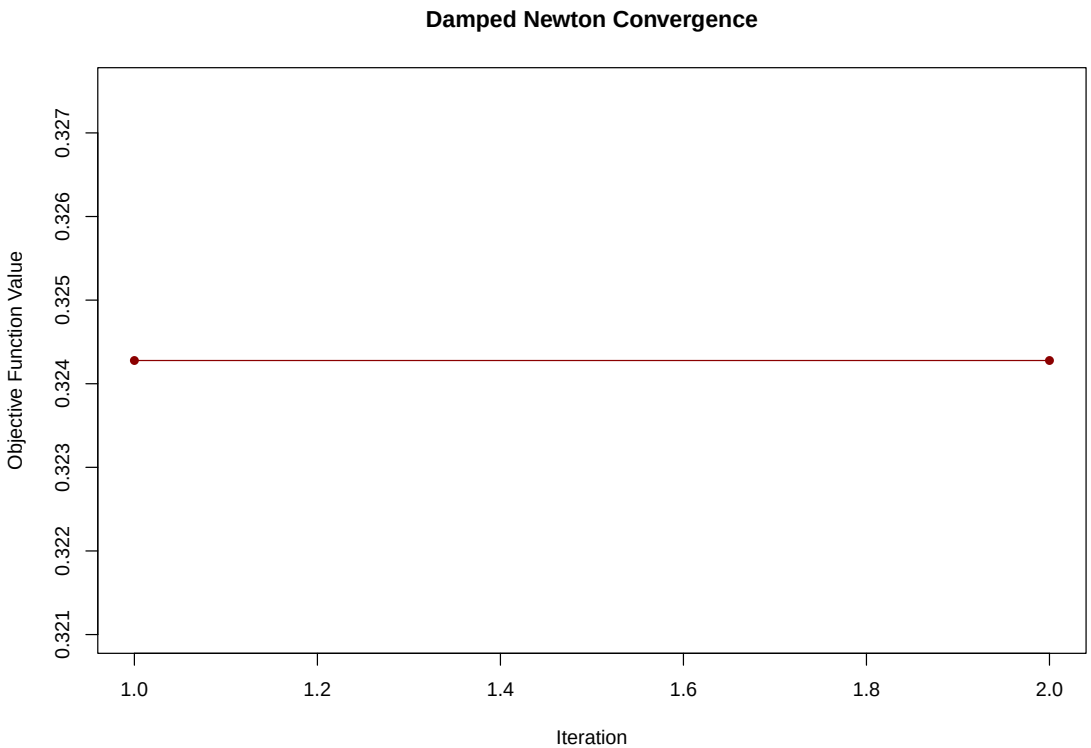

Figure 3: $\|\beta_{\text{true}} - \beta_{\text{GD}}\|_2 = 0.72$

Illustration - IV

Same p and n, but increase CI≈ 5000

The results based on backtracking line search for GD and damped Newton are shown next

Illustration - V

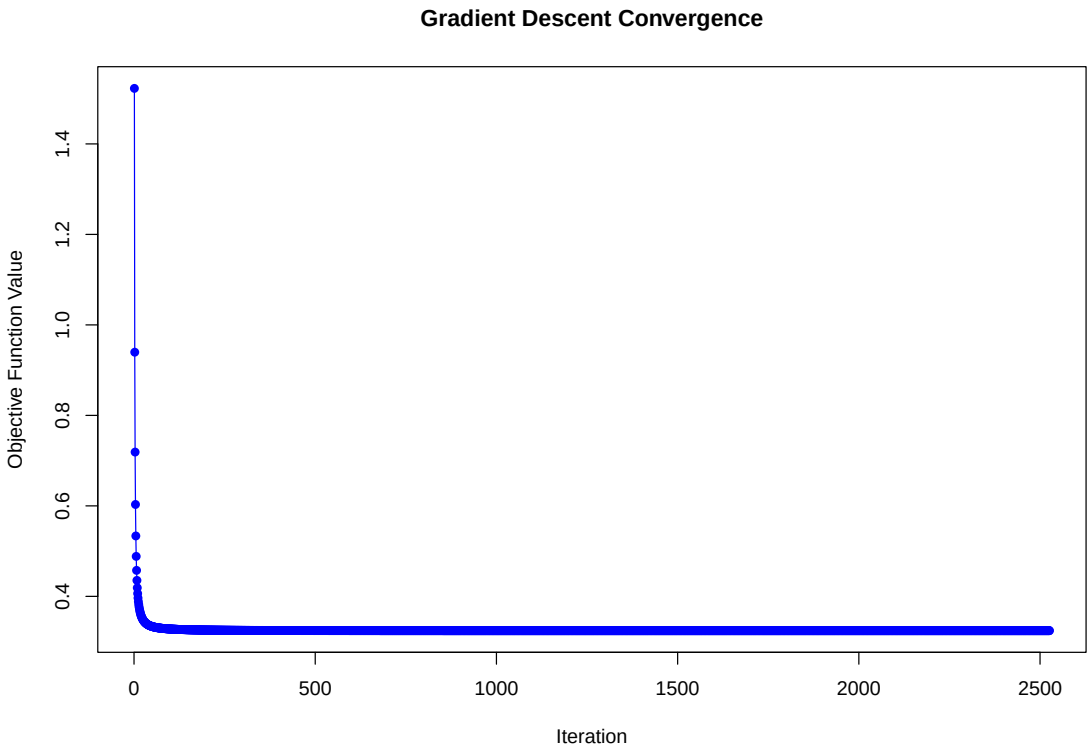

Figure 4: $\|\beta_{\text{true}} - \beta_{\text{GD}}\|_2 = 0.18$

Illustration - VI

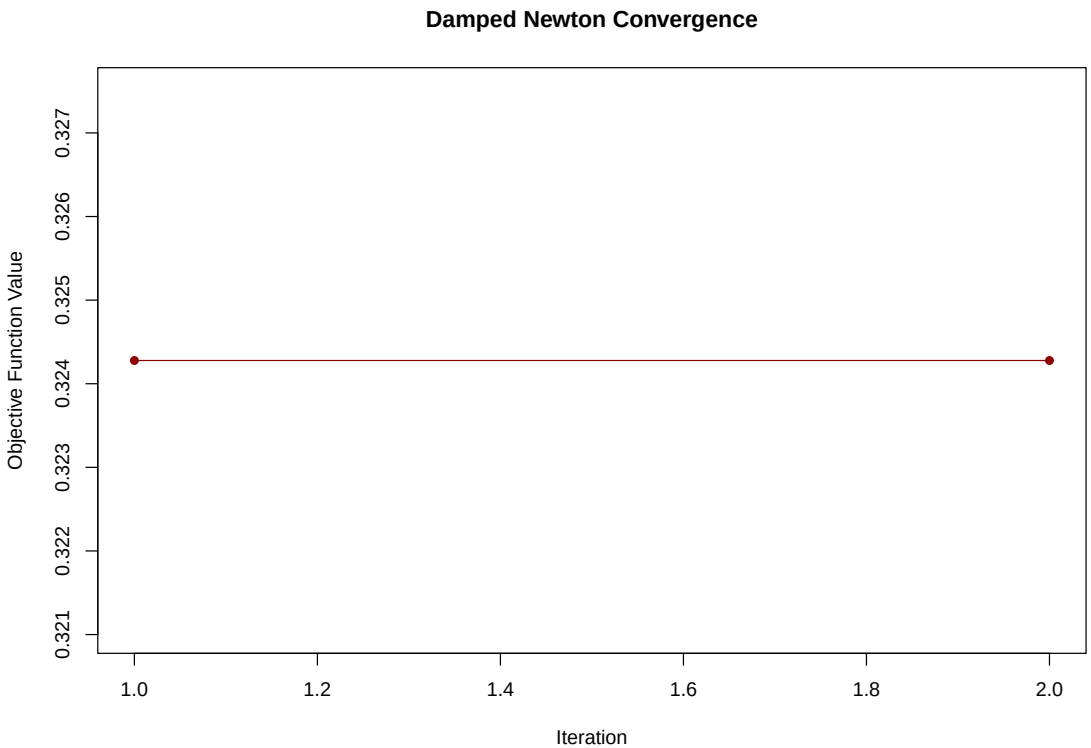

Figure 5: $\|\beta_{\text{true}} - \beta_{\text{GD}}\|_2 = 0.31$

Remarks

As expected, since the SSE function is a quadratic, Newton's algorithm converges essentially in a single iteration

However, as the condition number increases $(X^\top X)^{-1}$ becomes numerically unstable, which may compromise the accuracy of the regression coefficients estimated by Newton's algorithm

On the other hand, GD requires a very large number of iterations

A Numerical Illustration of Convergence of the Newton Algorithm for a non-quadratic function

Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by

$$f(x_1, x_2) = (10x_1^2 + x_2^2)/2 + 5\log\big(1 + \exp(-x_1 - x_2)\big)$$

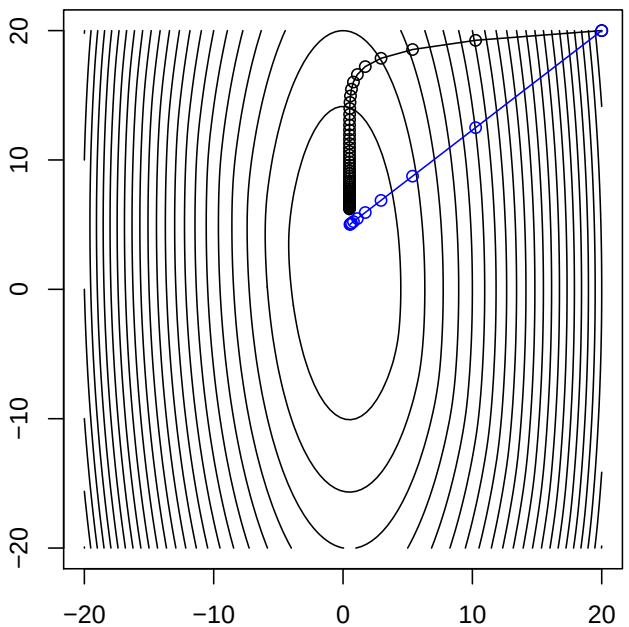

Figure 6: Iterates of GD (black) vs Newton with approximately equivalent step sizes

Variants and Fixes of Newton's Algorithm

There are two broad issues with the Newton algorithm

1. Expensive to calculate the update, due to the need to calculate inverse of the Hessian matrix
2. What if the Hessian is not strictly positive definite?
   Or the Hessian is ill-conditioned (i.e., the ratio of the maximum to the minimum of its eigenvalues is extremely large, so that the Hessian is close to being singular)?

Variants and Fixes of Newton's Algorithm

There are two broad issues with the Newton algorithm

1. Expensive to calculate the update, due to the need to calculate inverse of the Hessian matrix
2. What if the Hessian is not strictly positive definite?
   Or the Hessian is ill-conditioned (i.e., the ratio of the maximum to the minimum of its eigenvalues is extremely large, so that the Hessian is close to being singular)?

We address both issues next; first issue 2 and then issue 1

The Levenberg-Marquardt Algorithm - I

The idea behind this modification of the Newton algorithm is to make the latter applicable (namely to use the direction $d_k = -\left[\nabla^2 f(x_k)\right]^{-1}\nabla f(x_k)$), when the Hessian is NOT positive definite

To fix this issue, use instead the following direction

$$\tilde{d}_k = -\left[\nabla^2 f(x_k) + \mu_k I\right]^{-1}\nabla f(x_k), \tag{6}$$

for an appropriate $\mu_k > 0$

The Levenberg-Marquardt Algorithm - II

Note that:

1. As $\mu_k \to 0$, the Levenberg-Marquardt algorithm "becomes" the Newton algorithm
2. As $\mu_k \to \infty$, the Levenberg-Marquardt algorithm "becomes" the GD algorithm with a very small step size

To see why the last statement is valid, note that for very large values of μ_k, we have $(H + \mu_k I) \approx \mu_k I$ and therefore the Levenberg-Marquardt update becomes approximately

$$x_{k+1} = x_k - \frac{1}{\mu_k} \nabla f(x_k) = \text{ a GD update!!} \tag{7}$$

In practical implementations of the Levenberg-Marquardt algorithm, one starts with a small value of μ_k and increase it slowly until $f(x_{k+1}) < f(x_k)$ is observed (descent property)

Reducing the Computational Cost of Newton's Algorithm

The standard descent direction involves taking the inverse of the $n \times n$ Hessian matrix, which can become very expensive for very large n (e.g., in the thousands)

Several proposals are available in the literature:

1. Let $H_0 \equiv \nabla^2 f(x_0)$; i.e., the Hessian at initial point x_0
   Then, use the following update rule

$$x_{k+1} = x_k - \eta_k H_0^{-1} \nabla f(x_k) \tag{8}$$

   A variant is to update the Hessian every ℓ iterations (say $\ell = 5, 10$) to reduce the computational cost somewhat

2. Let $\tilde{H}_k \equiv \text{diag}\left(\frac{\partial^2 f(x_k)}{(\partial x_i)^2}\right)_{i=1}^n$; i.e., calculate only the diagonal elements of the Hessian at x_k
   Then, use the following update rule

$$x_{k+1} = x_k - \eta_k \tilde{H}_k^{-1} \nabla f(x_k) \tag{9}$$

3. Quasi-Newton methods aiming to approximate the Hessian (outside the scope of STAT 102B)