IE521 - Homework 5

Abhishek Avinash Narwekar
an41@illinois.edu

1 Logistic Regression

The unconstrained optimization problem we are trying to solve is:

$$\min_w \quad f(w) := \sum_{i=1}^{m} \log(1 + e^{-y_i w^T x_i}) + \frac{\lambda}{2}||w||^2$$

The gradient and hessian for this program are:

$$\nabla f(w) = \sum_{i=1}^{m} \frac{-y_i e^{-y_i w^T x_i} x_i}{1 + e^{-y_i w^T x_i}}$$

$$\nabla^2 f(w) = \sum_{i=1}^{m} \frac{e^{-y_i w^T x_i} x_i x_i^T}{\left(1 + e^{-y_i w^T x_i}\right)^2} + \lambda \mathbf{I}$$

Additionally, we set $L = \frac{1}{4}\lambda_{max}(X^T X) + \lambda$ and $\gamma = \frac{1}{L}$.

Results on WDBC Dataset

The functions `gradient_descent()` and `Newton()` have been written in HW5.py. The file also depends on the WDBC dataset files. Note that the variable name `lambda()` is reserved in Python; thus, we use `lambda_` instead.

Figure 1 plots the variation of the objective function's trajectory upon varying the step size in gradient descent. For Newton method, we initialize the weight vector with zero-mean, but with different variance. Figure 1 shows the effect of the standard deviation on the trajectories. As we can see, the trajectories are rather similar for all the initializations. Also, the trajectories diverge upon initialization with a large non-zero mean, such as 1, or a relatively large standard deviation, such as 0.01.

And finally, we compare the gradient descent and the Newton methods. I ran the gradient descent for over 10000 iterations, without observing convergence. The Newton method, however, converges in just 10 iterations, thereby demonstrating the efficacy of Newton method.

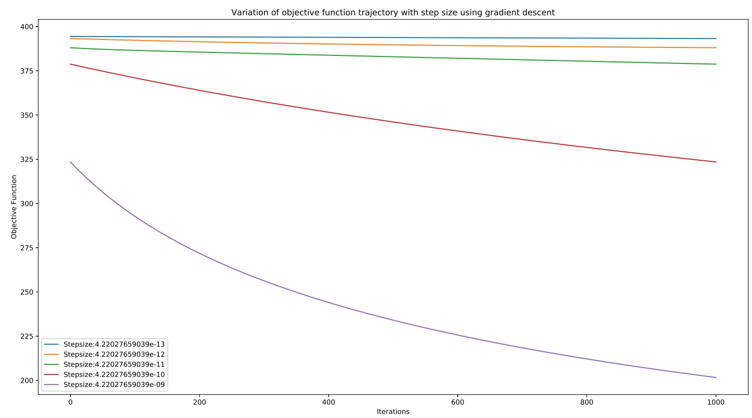

Figure 1: Varition of objective function trajectory with step size for gradient descent

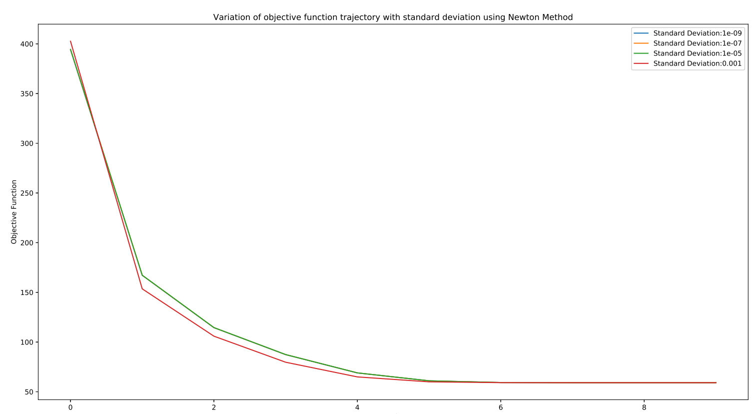

Figure 2: Varition of objective function trajectory with initialization for Newton Method

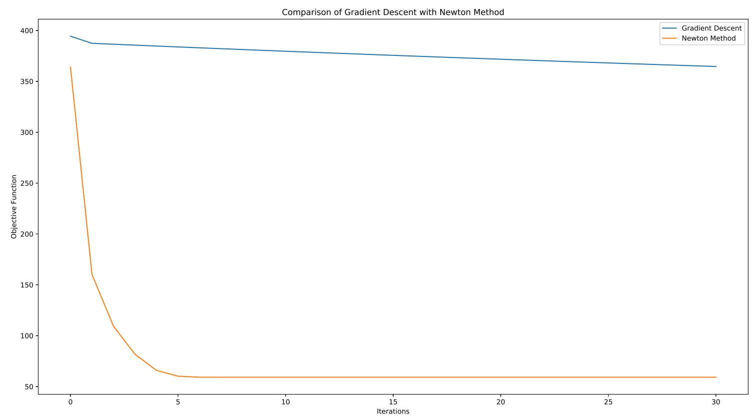

Figure 3: Comparison of trajectories in gradient descent and Newton Method

2 Linear Program

The unconstrained self-concordant minimization problem is as follows:

$$\min_x f(x) := \quad c^T x - \sum_{k=1}^{m} \log(b_k - a_k^T x)$$

The gradient and the hessian for this program are:

$$\nabla f(x) = c + \sum_{k=1}^{m} \frac{a_k}{b_k - a_k^T x}$$
$$\nabla^2 f(x) = \sum_{k=1}^{m} \frac{a_k a_k^T}{(b_k - a_k^T x)^2}$$

The update equation is:

$$x_{t+1} = x_t - \frac{1}{1+\lambda_t}[\nabla^2 f(x_t)]^{-1} \nabla f(x_t)$$

where $\lambda_t = \sqrt{\nabla f(x_t)^T [\nabla^2 f(x_t)]^{-1} \nabla f(x_t)}$.

Result on Artificial Dataset

The `damped_Newton()` has been implemented in HW5.py. Here too, the input parameter has been renamed to `lambda_`.

For the dataset described in the problem, we first find the approximate solution. We then study how the function approaches this approximate solution.

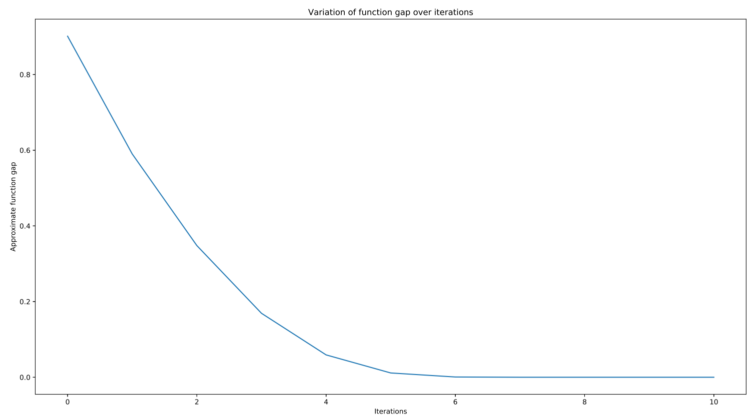

Figure 4: Function gap as a function of iterations

The function gap is shown in figure 4. The variation of the Newton decrement as a function of iterations while doing so is shown in figure 5.

It must be noted, however, that the damped Newton method **diverges** for certain initializations of A, b, c.

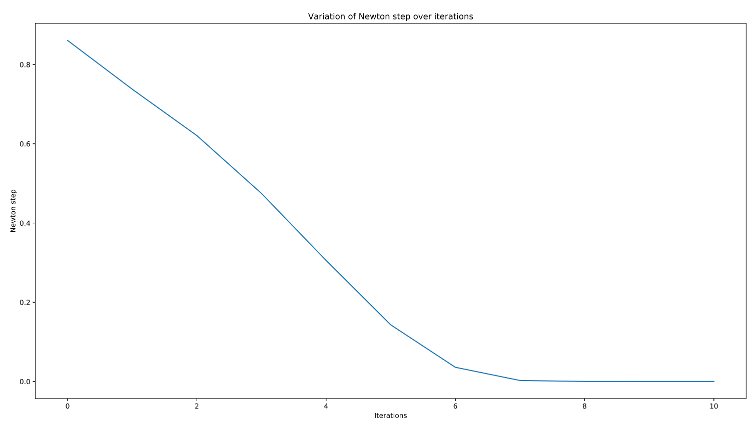

Figure 5: Variation of Newton Decrement as a function of iterations