Project: Regularized Newton methods for machine learning

Introduction

This project is concerned with going beyond first-order techniques in machine learning. Algorithmic frameworks such as gradient descent and stochastic gradient are inherently first-order methods, in that they rely solely on first-order derivatives. **Second-order methods**, on the other hand, make use of higher-order information, either explicitly or implicitly. Although those techniques are widely used in scientific computing, their use in machine learning has yet to be generalized. The goal of this project is to illustrate the performance of these techniques on learning problems involving both synthetic and real data.

The project is divided in three parts. In Section 1, we introduce Newton's method. We then describe a globally convergent version of this method in Section 2. Finally, we connect Newton-type methods to the machine learning setting by introducing subsampling Newton methods: this is the purpose of Section 3.

Implementation guidelines

- All Python libraries are allowed, but the project should **not rely on existing implementations of optimization methods**.
- NumPy structures and numerical procedures from the `scipy` library (except optimization ones!) are recommended, especially for the linear algebra calculation (matrix inverse, eigenvalues) but not mandatory.

Question guidelines

- A comparison between methods consists in reporting numerical results (number of iterations, final function values, convergence plots) and commenting on them. *Is there a clear winner in the comparison? Are there results that are surprising to you?*
- When details of the implementation are left open, you should choose settings that allow for fast convergence, or that look informative to you. Your comments should reflect these choices.
- The goal of testing several values of an hyperparameter (e.g. a constant step size) is to assess the robustness of a given method with respect to this hyperparameter. *Are the results sensitive to changes in the value of the hyperparameter? Can you identify regimes of values that yield similar results (such as the large batch and mini-batch regimes for the batch size in stochastic gradient)?*

1 Newton's method

The purpose of this section is to present the key method this project is based upon, which will be implemented and tested on small-dimensional toy problems of the generic form

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}), \tag{1}$$

where f is a twice continuously differentiable function (which we denote by $f \in \mathcal{C}^2$).

Newton's method is an optimization algorithm based on using the first- and second-order derivatives of the objective function. Starting from a point $\boldsymbol{w}_0 \in \mathbb{R}^d$, the method generates a sequence $\{\boldsymbol{w}_k\}_k$ from the following recursion:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \boldsymbol{d}_k, \quad \boldsymbol{d}_k = -\left[\nabla^2 f(\boldsymbol{w}_k)\right]^{-1} \nabla f(\boldsymbol{w}_k). \tag{2}$$

One immediately notices that iteration (2) is not well-defined, since the Hessian matrix of f at $\boldsymbol{w}_k$ may not be invertible. On the other hand, note that when the function is strongly convex, we have $\nabla^2 f(\boldsymbol{w}) \succ \mathbf{0}$ for any $\boldsymbol{w} \in \mathbb{R}^d$, and the Newton iteration is well defined.

One of the main characteristics of Newton's method is its fast local convergence rate guarantees. When f is a strongly convex quadratic, Newton's method converges in one iteration. If f is strongly convex, it can be shown that Newton's method converges at a local quadratic rate, i.e. if $\boldsymbol{w}_0$ is close enough to the global minimum $\boldsymbol{w}^*$ of f, we have:

$$\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\| \leq C \|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2,$$

where $C > 0$. By comparison, with a standard gradient descent approach, one would obtain a result of the form $\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\| \leq C \|\boldsymbol{w}_k - \boldsymbol{w}^*\|$, i. e. a linear rate of convergence.

Implementation 1.1 *Implement Newton's method in its basic form (2). Your code should take f, ∇f and $\nabla^2 f$ as inputs.*

Question 1.1 *Suppose that we apply Newton's method to the quadratic problem:*

$$\underset{\boldsymbol{w}\in\mathbb{R}^3}{\textit{minimize}}\; q(\boldsymbol{w}) := (w_1 + w_2 + w_3 - 5)^2 + 3(w_1 - w_2)^2 + 2(w_2 - 2\,w_3)^2. \tag{3}$$

This problem is a strongly convex quadratic function, with a unique minimizer $\boldsymbol{w}^* = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$.

a) Write down the first iteration of Newton's method for problem (3), and show that Newton's method indeed converges in one iteration to the solution.

b) Run the method starting from the origin and two other starting points of your choice. Do you indeed observe convergence in one iteration?

Question 1.2 *We now consider the celebrated Rosenbrock function*

$$\underset{\boldsymbol{w}\in\mathbb{R}^2}{\textit{minimize}}\; 100(w_2 - w_1^2)^2 + (1 - w_1)^2. \tag{4}$$

a) Apply your implementation of Newton's method to this problem starting from

$$\boldsymbol{w}_{01} := \begin{bmatrix} -1.2 \\ 1 \end{bmatrix}, \quad \textit{and} \quad \boldsymbol{w}_{02} := \begin{bmatrix} 0 \\ \frac{1}{400} + 10^{-12} \end{bmatrix}.$$

Report your results and your observations. Does the method converge?

b) Could we run the method starting from the point $\boldsymbol{w}_{03} = \begin{bmatrix} 0 \\ 0.005 \end{bmatrix}$*? How does that illustrate that Newton's method is local in nature?*

2 A globally convergent version of Newton's method

Newton's method comes with strong local guarantees, but not global ones. For general nonconvex functions (or even convex functions that are not strongly convex), the choice of the initial point determines whether or not the method will converge, or be well-defined. *Globalization techniques* were developed to guarantee that Newton's method would converge independently of its starting point, and regardless of the convexity of the problem. In this project, we will focus on one globalization technique based on regularization.

Suppose that we are at iteration k of the method: in order to apply a Newton-like iteration, one can always select a nonnegative value γ_k such that $\nabla^2 f(\boldsymbol{w}_k) + \gamma_k \boldsymbol{I}_d \succ 0$ and compute a Newton-type direction by replacing the Hessian by the matrix $\nabla^2 f(\boldsymbol{w}_k) + \gamma_k \boldsymbol{I}_d$. As a result, we obtain the following Newton-type iteration:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \boldsymbol{d}_k, \quad \boldsymbol{d}_k = -[\nabla^2 f(\boldsymbol{w}_k) + \gamma_k \boldsymbol{I}]^{-1} \nabla f(\boldsymbol{w}_k). \tag{5}$$

This process is called quadratic regularization, since it corresponds to adding a quadratic to the objective. Provided γ_k is large enough, the direction above will always be well-defined: we seek values of γ_k that lead to a decrease in the objective function.

Algorithm 1 describes an adaptive way of computing γ_k, akin to a line-search strategy for computing stepsizes. The initial choice for γ_k is made so that the direction $\boldsymbol{d}_k$ and all subsequent directions are well defined. Indeed, any value tried out by the algorithm will be such that

$$\lambda_{\min}\left(\nabla^2 f(\boldsymbol{w}_k) + \gamma_k \boldsymbol{I}\right) > 0,$$

where we recall that $\lambda_{\min}(\boldsymbol{A})$ stands for the minimum eigenvalue of the matrix $\boldsymbol{A}$.

Algorithm 1: Iteration k of Newton's method with quadratic regularization.

Inputs: $k \in \mathbb{N}$, $\boldsymbol{w}_k \in \mathbb{R}^d$, $c \in (0,1)$, $\mu > 1$.
Compute $\gamma_k = \mu \max\left\{-\lambda_{\min}(\nabla^2 f(\boldsymbol{w}_k)), 10^{-10}\right\}$.
Compute $\boldsymbol{d}_k$ using the formula in (5).
while $f(\boldsymbol{w}_k + \boldsymbol{d}_k) \geq f(\boldsymbol{w}_k) + c\boldsymbol{d}_k^{\mathrm{T}} \nabla f(\boldsymbol{w}_k)$ **do**
    Set $\gamma_k = \mu \gamma_k$.
    Recompute $\boldsymbol{d}_k$ using the formula in (5) and the new value of γ_k.
end

Under classical assumptions, it is possible to show convergence of this framework regardless of the starting point: this method is thus *globally convergent*. Under some additional conditions, it is also possible to obtain local convergence results, that are usually worse than that of Newton's method because of the use of γ_k. Still, regularized Newton methods are widely used in nonlinear optimization.

Implementation 2.1 *Implement the globalized version of Newton's method using the procedure described in Algorithm 1.*

Question 2.1 *Apply Newton's method with regularization to the problem (4) using $c = 0.0001$ and $\mu = 2$, as well the two initial points mentioned in Question 1.2. Compare your results with those obtained for the basic Newton iteration.*

Question 2.2 *Try out a few values for c and μ (report the results of your tests). How sensitive does the method appear to be to these values?*

3 Subsampling Newton-type methods

The goal of this section is to develop a practical variant of Newton's method that is tailored to machine learning formulations. For the purpose of this project, we will follow the lectures and focus on finite-sum problems of the form

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}), \tag{6}$$

where every f_i is $\mathcal{C}^2$ and depends on one data point. Our canonical example will be the logistic regression problem used in the course notebooks, which is strongly convex.

Exemple 3.1 *Given a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, we consider the problem*

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\textit{minimize}}\, f(\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}), \qquad f_i(\boldsymbol{w}) = \ln(1+\exp(-y_i \boldsymbol{x}_i^T \boldsymbol{w})) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2. \tag{7}$$

with $\lambda > 0$. Every function f_i is $\mathcal{C}^2$, and we have

$$\nabla f_i(\boldsymbol{w}) = -\frac{y_i}{1+\exp(y_i \boldsymbol{x}_i^T \boldsymbol{w})}\boldsymbol{x}_i + \lambda \boldsymbol{w}. \tag{8}$$

as well as

$$\nabla^2 f_i(\boldsymbol{w}) := \frac{\exp(y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i)}{\left(1+\exp(y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i)\right)^2}\boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} + \lambda \boldsymbol{I}_d. \tag{9}$$

In subsampling Newton methods, we construct stochastic gradient and Hessian estimates from batches of data points, akin to a batch stochastic gradient approach. At every iteration k of the algorithm, given the current iterate $\boldsymbol{w}_k$, one computes subsampling derivatives as

$$\nabla f_{\mathcal{S}_k}(\boldsymbol{w}_k) = \frac{1}{|\mathcal{S}_k|}\sum_{i\in\mathcal{S}_k} \nabla f_i(\boldsymbol{w}_k), \quad \nabla^2 f_{\mathcal{S}_k^H}(\boldsymbol{w}_k) = \frac{1}{|\mathcal{S}_k^H|}\sum_{i\in\mathcal{S}_k^H} \nabla^2 f_i(\boldsymbol{w}_k), \tag{10}$$

where $\mathcal{S}_k$ and $\mathcal{S}_k^H$ are sets of random indices drawn in $\{1, \ldots, n\}$. The subsampling iteration then reads

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \boldsymbol{d}_k, \quad \boldsymbol{d}_k = -\left[\nabla^2 f_{\mathcal{S}_k^H}(\boldsymbol{w}_k) + \gamma_k \boldsymbol{I}\right]^{-1} \nabla f_{\mathcal{S}_k}(\boldsymbol{w}_k), \tag{11}$$

where $\gamma_k > 0$ guarantees that the update is well defined. Since the algorithm now relies on subsampling, we can no longer rely on checking decrease for the full objective function f. A natural proxy for this function consists in using the same sample set than that used for the gradient approximation: this is the procedure described in Algorithm 2.

Algorithm 2: Iteration k of Newton's method with quadratic regularization and subsampling.

Inputs: $k \in \mathbb{N}$, $\boldsymbol{w}_k \in \mathbb{R}^d$, $\mathcal{S}_k$, $\mathcal{S}_k^H$, $c \in (0, 1)$, $\mu > 1$.
Compute $\gamma_k = \mu \max\left\{-\lambda_{\min}(\nabla^2 f_{\mathcal{S}_k^H}(\boldsymbol{w}_k)), 10^{-10}\right\}$.
Compute $\boldsymbol{d}_k$ using the formula in (11).
while $f_{\mathcal{S}_k}(\boldsymbol{w}_k + \boldsymbol{d}_k) \geq f_{\mathcal{S}_k}(\boldsymbol{w}_k) + c\boldsymbol{d}_k^{\mathrm{T}} \nabla f_{\mathcal{S}_k}(\boldsymbol{w}_k)$ **do**
    Set $\gamma_k = \mu \gamma_k$.
    Recompute $\boldsymbol{d}_k$ using the formula in (11) with the new value of γ_k.
end

Implementation 3.1 *Implement a subsampling Newton method with the following requirements:*

- *The method should take $|\mathcal{S}_k|$ and $|\mathcal{S}_k^H|$ as inputs (for simplicity, we only consider constant sample sizes).*
- *The method should use Algorithm 2 to compute $\boldsymbol{d}_k$ at every iteration.*

Question 3.1 *Using the same (synthetic) dataset than in lab 3 of the course (on stochastic gradient methods), compare the subsampling Newton method with the regularized Newton method of Section 2. The comparison should satisfy the following requirements:*

- *Algorithms 1 and 2 should be employed for regularized Newton and subsampling Newton, respectively, with the same values for c and μ.*
- *The comparison should be conducted in terms of epochs, i.e. the results should be plotted as a function of the number of accesses to data points.*
- *The comparison should include several variants of subsampling Newton based on varying $|\mathcal{S}_k|$ and $|\mathcal{S}_k^H|$, with at least three different values for $|\mathcal{S}_k|$ and two values for $|\mathcal{S}_k^H|$.*
- *The variant corresponding to $|\mathcal{S}_k| = |\mathcal{S}_k^H| = 1$ must appear in the comparison.*

Comment on the obtained results, and the sensitivity of the method to the sample sizes.

Question 3.2 *Compare the best subsampling Newton variant you obtained on the problem with batch stochastic gradient using $\max\{|\mathcal{S}_k|, |\mathcal{S}_k^H|\}$ as a batch size, where $\mathcal{S}_k$ and $\mathcal{S}_k^H$ are the values considered in the subsampling Newton method. Use a constant stepsize for stochastic gradient proportional to $\frac{1}{L}$, where $L = \frac{4\|\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\|}{n} + \lambda$ is the Lipschitz constant of ∇f.*