Computer Science Department, Ben Gurion University of the Negev

Optimization methods with Applications - Spring 2020

Home Assignment 3

Due on June 4th.

May 29, 2020

1. **Convexity.**

   (a) For each of the following functions, write if they are convex, concave, or none of those. If they are, prove it using one of the definition seen in class.

   i. e^{ax}

   ii. $-log(x)$

   iii. $log(x)$

   iv. $|x|^a, a \geq 1$

   v. x^3

   (b) Let $f(x) = x^T Ax + b^T x + c$. Is $f(x)$ convex? If yes, under which conditions?

   (c) Prove the following theorem: Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable in a convex domain Ω. Then, the following are equivalent:

   i. f is convex

   ii. $f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad , \forall x, y \in \Omega$

   Guidance for i $\Rightarrow$ ii: From convexity show that for every $0 < \theta < 1$ you have

$$f(x) + \theta(f(y) - f(x)) \geq f(x + \theta(y - x)).$$

   Furthermore, for $\theta \to 0$ you have:

$$f(x + \theta(y - x)) = f(x) + \theta \nabla f(x)^T (y - x).$$

   Using these two you can eliminate θ and reach the desired conclusion. Note that this has a geometrical interpretation, and is not just a mathematical trick. It's better to understand why.

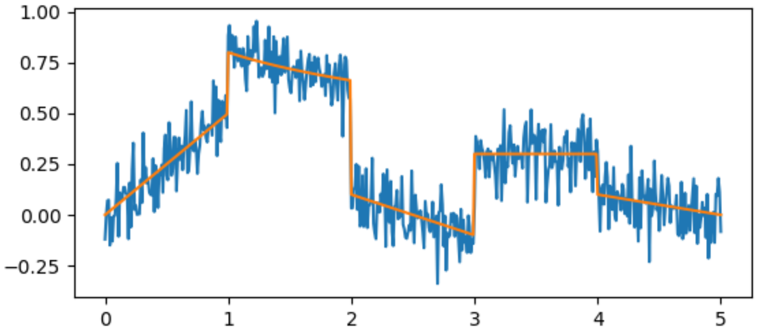

Figure 1: the clean and noisy signals.

2. **Iterative re-weighted least squares (IRLS)**

Introduction: There are many cases in real life applications where we wish to solve problems that we don't really know how to handle efficiently. One such example is a minimization of a linear equation in ℓ_p for $p \neq 2$:

$$\|A\mathbf{x} - \mathbf{b}\|_p^p. \tag{1}$$

The method of "Iterated Re-weighted Least Squares" (IRLS) states that instead of solving a problem like (1) directly, we solve a series of weighted LS problems by using the equality with a diagonal matrix W

$$\|\mathbf{v}\|_p^p = \mathbf{v}^T W \mathbf{v} = \|\mathbf{v}\|_W^2 \text{ where } W_{ii} = |v_i|^{p-2}. \tag{2}$$

To solve (1) using IRLS, we will define the series

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_{W^{(k)}}^2, \tag{3}$$

where $(W^{(k)})_{ii} = |r_i^{(k)}|^{p-2}$, with $\mathbf{r}^{(k)} = A\mathbf{x}^{(k)} - \mathbf{b}$ is the residual. Note that $W^{(k)}$ depends on $\mathbf{x}^{(k)}$, but is kept fixed in the minimization (3) to allow us to have a closed form solution. That is the reason that a few iterations of IRLS are needed.

1D Total Variation using IRLS: One such popular example for using IRLS is the total variation, which we will solve in 1D. Assume the piece-wise smooth function

$$f(x) = \begin{cases} 0.5x & 0 \leq x < 1 \\ 0.8 - 0.2\log_2(x) & 1 \leq x < 2 \\ 0.7 - 0.2x & 2 \leq x < 3 \\ 0.3 & 3 \leq x < 4 \\ 0.5 - 0.1x & 4 \leq x \leq 5 \end{cases}$$

Let $\mathbf{f}$ be a sampling of $f(x)$ on some grid of $n+1$ nodes, and let $\mathbf{y}$ be a noisy version of it, with additive white noise of std 0.1: $\mathbf{y} = \mathbf{f} + \eta$; $\eta \sim \mathcal{N}(0, 0.1^2 I)$. Figure 1 shows the two signals.

We wish to recover the clean signal $\mathbf{f}$ given the noisy signal $\mathbf{y}$ while preserving the sharp jumps in $\mathbf{f}$. To this end, we try to solve

$$\arg\min_{\mathbf{x}} \|\mathbf{x}-\mathbf{y}\|_2^2 + \lambda \sum_{i=1}^{n-1} |x_{i+1} - x_i| = \arg\min_{\mathbf{x}} \|\mathbf{x}-\mathbf{y}\|_2^2 + \lambda \|G\mathbf{x}\|_1, \tag{4}$$

where G is known to be the discrete gradient matrix. For n=5, for example:

$$G = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Using the ℓ_1 here preserves edges in signals, or images in the case of 2D. Since the ℓ_1 norm is a bit tricky to handle here (more later in the course...) one way to handle this is by IRLS.

(a) For comparison later, solve problem (4) with ℓ_2 instead of ℓ_1 (that is use $\frac{\lambda}{2}\|G\mathbf{x}\|_2^2$ instead of $\lambda\|G\mathbf{x}\|_1$), using a single non-weighted LS solve. Use $\lambda = 80$, and plot the solution. You can try other values too.

(b) Now run 10 IRLS iterations, starting from $W^{(0)} = I$, and using the approximation

$$(W^{(k)})_{ii} = \frac{1}{|(G\mathbf{x}^{(k)})_i| + \epsilon}.$$

Use $\lambda = 1$, and $\epsilon = 0.001$. Verify that indeed the objective in (4) is reduced. Plot the final result.

(c) (No need to submit anything) Observe how the TV approximation preserves the edges better than the ℓ_2 smoothing in (a). Note that TV is designed to promote piece-wise linear functions and not piece-wise smooth functions as we gave it.

A Julia code for generating the experiments:

```
x = collect(0:0.01:5);
n = length(x);
one = div(n-1,5); # = 100
f = zeros(size(x))
f[1:one] = 0.0 .+ 0.5.*x[1:one];
```

```
f[(one+1):2*one] = 0.8 .- 0.2.*log.(x[101:200]);
f[(2*one+1):3*one] = 0.7 .- 0.3.*x[(2*one+1):3*one];
f[(3*one+1):4*one] .= 0.3;
f[(4*one+1):(5*one+1)] .= 0.5 .- 0.1.*x[(4*one+1):(5*one+1)];
G = spdiagm(-1=>-ones(n-1),0=>ones(n));
G = G[2:end,1:end]
etta = 0.1*randn(size(x));
y = f .+ etta
plot(x,y);plot(x,f)
```

3. **Non-linear least squares data fitting using Gauss Newton or Levenberg–Marquardt**.

Introduction In many scenarios we wish to fit given data to a model using some parameters. We already know that Least Squares is a popular approach for penalizing for the model error. However, what if the model is non-linear?

Suppose that you are given with (possibly noisy) observations $\mathbf{y}^{obs}$ and you wish to fit a model $\mathbf{f}(\theta)$ to this data, for prediction or analysis. θ is a vector of model parameters. A popular choice would be to solve

$$\min_{\theta} \frac{1}{2}\|\mathbf{f}(\theta) - \mathbf{y}^{obs}\|_2^2 \tag{5}$$

which may be accompanied with some regularization over θ, which we will ignore here. Denote $F(\theta) = \frac{1}{2}\|\mathbf{f}(\theta) - \mathbf{y}^{obs}\|_2^2$

A popular approach to solve the problem above is by some descent algorithm. For example the steepest descent method would yield:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)}\nabla_\theta F(\theta^{(k)})$$

This method, however, often converges slowly, and using a higher-order method is beneficial. The method of Gauss-Newton states to approximate $\mathbf{f}$ up to first order

$$\mathbf{f}(\theta^{(k)} + \mathbf{d}) \approx \mathbf{f}(\theta^{(k)}) + \mathbf{J}(\theta^{(k)})\mathbf{d}$$

where $\mathbf{J}(\theta^{(k)})$ is the Jacobian of $\mathbf{f}$ w.r.t θ, and $\mathbf{d}$ is a perturbation in θ. Using this approximation, we find the search direction by minimizing the resulting approximation:

$$\mathbf{d}_{GN}^{(k)} = \arg\min_{\mathbf{d}} \frac{1}{2}\|\mathbf{f}(\theta^{(k)}) + \mathbf{J}(\theta^{(k)})\mathbf{d} - \mathbf{y}^{obs}\|_2^2, \tag{6}$$

and the stepsize is obtained by linesearch:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha^{(k)}\mathbf{d}_{GN}^{(k)}.$$

(a) Show that the gradient of (5) is given by: $\nabla F(\theta) = \mathbf{J}^\top(\mathbf{f}(\theta) - \mathbf{y}^{obs})$, where $\mathbf{J}$ is the Jacobian mentioned above.

(b) Show that the solution of (6) is given by solving the linear system for $\mathbf{d}$:

$$\mathbf{J}^\top\mathbf{J}\mathbf{d} = -\nabla F(\theta^{(k)}).$$

This linear system can be solved using a factorization, or an iterative method like Conjugate Gradients or Gauss–Seidel.

(c) In some cases, the Jacobian $\mathbf{J}$ may be ill-conditioned or of low-rank, and the system in the previous section becomes unstable. A quick fix for this results in the Levenberg–Marquardt variant of GN:

$$\mathbf{d}_{LM}^{(k)} = \arg\min_{\mathbf{d}} \frac{1}{2}\|\mathbf{f}(\theta^{(k)}) + \mathbf{J}(\theta^{(k)})\mathbf{d} - \mathbf{y}^{obs}\|_2^2 + \frac{\mu}{2}\|\mathbf{d}\|_2^2, \tag{7}$$

for some parameter μ. Show the corresponding linear system that needs to be solved to get $\mathbf{d}_{LM}^{(k)}$.

(d) Explain why $\mathbf{d}_{LM}^{(k)}$ is guaranteed to be a descent direction.

(e) In this section we will solve a minimization to predict and analyze the number of COVID-19 confirmed cases in the USA. The data in `covid-19-USA.txt` (taken from `https://ourworldindata.org/coronavirus`) contains the number of confirmed cases in the USA from Jan 29th to Apr 28th - a total of 99 days. Consider the following simple Gaussian model:

$$\mathbf{f}(\theta) = \theta_1 \exp\left(-\theta_2(\mathbf{x} - \theta_3)^2\right),$$

where $\mathbf{x}$ is the vector of days (i.e., [1...99]). Find the parameters of the model that fit best with the given data in a least squares sense. Have two experiments: using Steepest Descent, and using Gauss-Newton or Levenberg–Marquardt. For each method, plot the convergence history graphs, that means showing the objective values $|F(\theta^{(k)}) - F(\theta^*)|$ in semilogarithmic scale in y. Compare and draw conclusions on the convergence of both options. Note that you don't have θ^* during the iterations, but only at the end. This way the plots go to zero.

You may start your minimization from: $\theta = [100000, 0.001, 110]$, which seem to be reasonable.

(f) Repeat the previous section for the scaled model

$$\hat{\mathbf{f}}(\hat{\theta}) = 1000000\hat{\theta}_1 \exp\left(-0.001\hat{\theta}_2(\mathbf{x} - 110\hat{\theta}_3)^2\right),$$

starting from $\hat{\theta} = [1, 1, 1]$. Is the convergence of both methods different? explain.

(g) (Not necessary for submission) Try to find a better model for prediction/data fit. You may find some information or inspiration at `https://www.cdc.gov/coronavirus/2019-ncov/covid-data/forecasting-us.html`.

4. **Non linear convex optimization for handwritten digits classification.**

   In this section we will examine the methods of Gradient Descent and Newton for the logistic regression problem for binary classification. Read Section 6.1 (Linear classifiers: Logistic regression) in the NOA lecture notes.

   (a) Write a function that, given data matrix $\mathbf{X}$ and labels, computes the logistic regression objective, its gradient, and its Hessian matrix (assume that the Hessian matrix is not so big).

   (b) Read Section 6.6 (Gradient and Jacobian verification), and verify that your code works. That is, verify that the gradient test passes, then the Hessian passes the Jacobian test with respect to the gradient (the Jacobian of the gradient vector is the Hessian). The tests can be applied using any data, even random.

   (c) We will now use the logistic regression model to classify handwritten digits from the MNIST data set.
   The data can be downloaded from: `http://yann.lecun.com/exdb/mnist/`
   Filter the digits from the data set and use the first 30000 instances from the dataset. You may use the attached Matlab files to filter the data.

   Run the following methods:

   - Gradient Descent.
   - Exact Newton.

   for the MNIST data sets for the digits 0/1 and 8/9. Show the convergence history: error values $|f(\mathbf{w}^{(k)}) - f(\mathbf{w}^*)|$ with respect to the training and testing data sets, in semilogarithmic scale in y.

   Explanation: the shift of $f(\mathbf{w}^*)$ aims just to make the train error series go to zero, for a more informative plotting. The convergence history for the testing (or validation) error, aims to verify that we're also decreasing the error for the test set. It may be that the test error will stop dropping at some point and start increasing: this is an indication of **overfitting**. For the test plots it may be that you need to define $\mathbf{w}^*$ as the point with minimum loss for the test case, and not the final converged answer.

   Note: the Hessian matrix may be singular. In such cases you can make it SPD by adding a scaled identity of your choice. Also, in this problem it is worthy to clip each of the weights $\mathbf{w}^{(k)}$ to be in $[-1, 1]$ at every iteration.

   (d) (Not for submission) Try the previous section with inexact Newton-CG, where the Newton problem is solved using a few iterations of Conjugate Gradients.