ECE 8823 (Convex Optimization), Spring 2017

Homework #4

Due Wednesday March 1, in class

Reading: B&V, Chapter 9

1. Using you class notes, prepare a 1-2 paragraph summary of what we talked about in class in the last week. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other things you have learned here or in other classes?). The more insight you give, the better.

2. (a) Implement Newton's algorithm: write MATLAB function

   ```
   function xstar = newton(f, gradf, hessf, x0, tol)
   ```

   that takes a function handle for evaluating a functional $f : \mathbb{R}^N \to R$, its gradient $\nabla f : \mathbb{R}^N \to \mathbb{R}^N$, and its Hessian $\nabla^2 f : \mathbb{R}^N \to S^N$, in addition to an initial point and a convergence tolerance. The algorithm should terminate when

   $$\lambda(\boldsymbol{x})^2/2 = \nabla f(\boldsymbol{x})^{\mathrm{T}}[\nabla^2 f(\boldsymbol{x})]^{-1}\nabla f(\boldsymbol{x})/2 \leq \epsilon,$$

   where ϵ is the tolerance prescribed above. There is pseudo-code in Section 9.5.2 of B&V that you can follow if you like. Turn in your code.

   (b) Compare and contrast Newton's method and gradient descent for solving Problem 7(c)–(e) on Homework 3. You might find the discussion in Appendix A.4.2 and A.4.4 in B&V useful in computing the gradient and Hessian. Turn in some representative plots along with your discussion.

   (c) Consider the problem of minimizing a functional of the form

   $$f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{c} \rangle - \sum_{m=1}^{M} \log(b_m - \langle \boldsymbol{x}, \boldsymbol{a}_m \rangle)$$

   on the open $\operatorname{dom} f = \{\boldsymbol{x} \ : \ \boldsymbol{A}\boldsymbol{x} < \boldsymbol{b}\}$. Compare and contrast gradient descent and Newton by considering random instances of this problem with $M = 500$, $N = 100$. If you make $\boldsymbol{b} > 0$, then the domain will be non-empty, and you can use $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ as a starting point. Make plots similar in spirit to Figure 9.6 and 9.21 in B&V (although don't worry about the exact line search — just make plots for backtracking).

3. (a) Implement the BFGS Quasi-Newton method: write MATLAB function

   ```
   function xstar = bfgs(f, gradf, x0, H0, tol)
   ```

   that takes a function handle for evaluating a functional $f : \mathbb{R}^N \to \mathbb{R}$, its gradient $\nabla f : \mathbb{R}^N \to \mathbb{R}^N$, in addition to an initial point, an initial symmetric positive definite matrix to use as the approximate Hessian at the first iteration, and a convergence tolerance. Turn in your code.

(b) Repeat part (b) of problem 2, comparing all three of gradient descent, Newton, and BFGS.

(c) Repeat part (c) of problem 2, comparing all three of gradient descent, Newton, and BFGS.

4. (a) Implement the Heavy Ball accelrated gradient descent method: write MATLAB function

```
function xstar = heavyball(f, gradf, alpha, beta, x0, tol)
```

that takes a function handle for evaluating a functional $f : \mathbb{R}^N \to R$, its gradient $\nabla f : \mathbb{R}^N \to \mathbb{R}^N$, and parameters $\alpha, \beta \in \mathbb{R}$, in addition to an initial point and a convergence tolerance. As for gradient descent, the algorithm should terminate when $\|\nabla f(\boldsymbol{x}^{(k)})\|_2^2 \leq \epsilon$, where ϵ is the tolerance prescribed above.
We will use fixed values of α and β, so you do not need to include any kind of line search.

(b) Test your code out by using it to solve random instances of the least-squares problem,

$$f(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2,$$

for some known $\boldsymbol{y} \in \mathbb{R}^M$ and $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, where we generate $\boldsymbol{A}$ and $\boldsymbol{y}$ using

```
M = 1000;
N = 800;
A = randn(M,N);
y = randn(M,1);
```

Us the origin as a starting point: $\boldsymbol{x}^{(0)} = \boldsymbol{0}$. In this case, a good value for the gradient multiplier is $\alpha = 1/(2\|\boldsymbol{A}\|^2)$, where $\|\boldsymbol{A}\|$ is the standard spectral norm (largest singular value) of $\boldsymbol{A}$. Play around to find values of β that work well; $\beta = 0.9$ works well for me. Report on the typical number of iterations needed for convergence. Here, a tolerance of 0.01 will be sufficient.

(c) Collect some statistics on the number of iterations required for convergence for both your heavy ball algorithm, and gradient descent with fixed step size (same α as above). Report on what you learn.

(d) For one particular instance of the problem, make a plot of $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{(k)}\|_2^2$ versus k for both the heavy ball and gradient descent methods. Comment on what you see.

5. In our discussions about the theory and practice of unconstrained optimization of differentiable functions, we often referred to the notion of a *descent direction*. Recall that $\boldsymbol{d}$ is a descent direction for f at a point $\boldsymbol{x}_0$ if

$$f(\boldsymbol{x}_0 + t\boldsymbol{d}) < f(\boldsymbol{x}_0) \quad \text{for some } t > 0.$$

For convex, differentiable f, this is equivalent to the condition $\langle \boldsymbol{d}, \nabla f(\boldsymbol{x}_0) \rangle < 0$.

(a) Prove that for a (not necessarily differentiable) convex function, the set

$$\mathcal{D}(\boldsymbol{x}_0) = \{\boldsymbol{d} : f(\boldsymbol{x}_0 + t\boldsymbol{d}) \leq f(\boldsymbol{x}_0) \text{ for some } \ t > 0\}$$

is a convex cone. $\mathcal{D}(\boldsymbol{x}_0)$ is called the *descent cone* or *cone of descent* of f at $\boldsymbol{x}_0$.

(b) Describe $\mathcal{D}(\boldsymbol{x}_0)$ for $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2$.

(c) Describe $\mathcal{D}(\boldsymbol{x}_0)$ for $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$.
(Your answer should be a very clean expression in terms of the set Γ_0 where $\boldsymbol{x}_0$ is non-zero and the signs of the entries of $\boldsymbol{x}_0$ on Γ_0.)

6. An *ellipsoid* is a set in $\mathbb{R}^N$ that has the form

$$\mathcal{E}_{\boldsymbol{P},\boldsymbol{b}} = \{\boldsymbol{x} \in \mathbb{R}^N : (\boldsymbol{x} - \boldsymbol{b})^{\mathrm{T}} \boldsymbol{P} (\boldsymbol{x} - \boldsymbol{b}) \leq 1\},$$

where $\boldsymbol{P}$ is a symmetric positive definite matrix, and $\boldsymbol{b}$ is a fixed vector. The ellipsoid is centered at $\boldsymbol{b}$, its axes (relative to $\boldsymbol{b}$) are the eigenvectors of $\boldsymbol{P}$, and the length of the axes are the square-roots of the invereses of the eigenvalues of $\boldsymbol{P}$. The volume of the ellipse, then, is proportional to the product-of-the-square-roots-of-the-eigenvalues, or more concisely $(\det \boldsymbol{P}^{-1})^{1/2}$.

The file `scatter_points.mat` contains a 2×100 matrix `X` whose columns are 100 different points in $\mathbb{R}^2$. You can look at them with

```
scatter(X(1,:), X(2,:))
```

Use CVX to find the $\boldsymbol{P}^\star, \boldsymbol{b}^\star$ that define the ellipse of minimal volume that contains all of the points. (Notice that minimizing $(\det \boldsymbol{P}^{-1})^{1/2}$ is the same as minimizing $\log\det \boldsymbol{P}^{-1}$.) Turn in your code, your answer for $\boldsymbol{P}^\star, \boldsymbol{b}^\star$, and the scatter plot of the points overlaid with the ellipse that you found.