ORIE 6300 Mathematical Programming I November 10, 2016

Problem Set 9

Due Date: November 17, 2016

Handwritten assignments will not be accepted

1. We saw in class that Lipschitz continuity of the gradient is key for ensuring convergence of the projected gradient method. In this exercise, you will learn more examples of functions with Lipschitz continuous gradients. Prove that the following functions have Lipschitz continuous gradients and provide a Lipschitz constant:

   (a) **Composition with linear.** Suppose that $f : \mathbb{R}^m \to \mathbb{R}$ is a function with an L-Lipschitz continuous gradient and let $A \in \mathbb{R}^{m\times n}$ be a matrix, and let b be a vector. Show that the function $f(Ax - b)$ has a Lipschitz continuous gradient. What is the its Lipschitz constant? (**Hint**: use the chain rule to compute the gradient.)

   (b) **Sums.** Suppose that $f_1, f_2 : \mathbb{R}^m \to \mathbb{R}$ are functions with L_1 and L_2-Lipschitz continuous gradients, respectively. Prove that the gradient of $f_1 + f_2$ is Lipschitz continuous, and determine its Lipschitz constant.

   (c) **Squared norm**: $f(x) = \frac{1}{2}\|x\|^2$.

   (d) **Logistic loss:** $f(x) = \sum_{i=1}^n \log(1 + \exp(x_i))$.

   (e) **Student's-t:** $f(x) = \sum_{i=1}^n \frac{\nu}{2} \log\left(1 + \frac{x_i^2}{\nu}\right)$ (where $\nu > 0$ is fixed).

   (f) sin**:** $f(x) = \sin(x)$.

   (g) **Hybrid norm:** $f(x) = \sum_{i=1}^n \sqrt{1 + x_i^2}$.

2. **Local Optimality.** Let $C \subseteq \mathbb{R}^n$ be a nonempty closed convex set. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. Let $\overline{x}$ be a local minimizer of f in C, i.e., there exists an $\delta > 0$ such

$$(\forall y \in C \cap B(\overline{x}; \delta)) \qquad f(y) \geq f(\overline{x}).$$

   Suppose further that f is differentiable at $\overline{x}$. Prove that $-\nabla f(\overline{x}) \in N_C(\overline{x})$.

3. **Robust Regression with Student's-t.** Consider the matlab script

```
%% Generate Data
clear all; close all; clc
step = .01;
grid = 0:step:1;
nb_samples = length(grid);
nb_corrupt = ceil(.5*nb_samples);
y_clean = sin(4*pi*grid)';
p = randperm(nb_samples);
```

```
y = y_clean;
y(p(1:nb_corrupt)) = y(p(1:nb_corrupt)) + 5*rand(nb_corrupt, 1);
y(1) = y_clean(1); y(end) = y_clean(end);% Make sure endpoints are clean


%% Set up model
rate = 9; % rate of sin
X = ones(nb_samples, 1);

for ii = 1:rate
    X = [X, sin(ii*pi*grid')];
end

%% Fit Least Squares
x_LS = X\y;

%% Solvers

params.max_iter = 500000;
params.tol = 1e-10;
params.init_x = x_LS; %% Initialize at the Least Squares Solution


params.X = X;
params.y = y;
nu = .01;
params.loss = @(r) f_ST(r, nu); %% Student's t
params.P_C = @(x) x; % unconstrained
[x_ST, history_ST] = projected_gradient( params );
D = pinv(X([1, nb_samples], :));
params.P_C = @(x) x - D*(X([1, nb_samples], :)*x - y([1, nb_samples])); % Constrained
[x_ST_constrained, history_ST_constrained] = projected_gradient( params );

%% Plot polynomial fit
figure;
scatter(grid, y);
hold on
plot(grid, y_clean);
plot(grid, X*x_LS);
plot(grid, X*x_ST);
plot(grid, X*x_ST_constrained);
legend('Noisy Signal', 'True signal', 'LS Fit', 'ST fit', 'x_ST_constrained');

%% Plot coefficients
figure;
```

```
plot(0:length(x_ST)-1, x_ST');
hold on
plot(0:length(x_ST)-1, x_ST_constrained');
legend('Unconstrained', 'Constrained');
```

and auxiliary function

```
function [f, g, Lip] = f_ST(r, nu)
f = sum(0.5*(nu).*log(1 + r.^2/nu));
if(nargout> 1)
    g = r.*nu./(nu + r.^2);
end
if(nargout > 2)
    Lip = 1;
end
end
```

(a) Fill in the `projected_gradient` function so that it solves the minimization problem:

$$\text{minimize}_{x\in C}\ \text{loss}(Xx - y),$$

where X_i is the ith row of X.

```
function [ x ] = projected_gradient( params )
max_iter = params.max_iter;
X = params.X;
y = params.y;
loss = params.loss;
P_C = params.P_C;
tol = params.tol;
x = params.init_x;
normX2 = norm(X)^2;
for ii = 1:max_iter

Fill in here

end
end
```

Expect `loss(X*x - y)` to return three arguments `[f, g, Lip]` where f is the numerical value of `loss(X*x - y)`, `g` is the gradient of `loss` at `X*x - y`, and `Lip` is the Lipschitz constant of ∇`loss`—in particular, you should expect `loss` to behave exactly like `f_ST`. Write your function so that it exits when $\|x^{k+1} - x^k\|/\gamma \leq tol$. **Include your code**. (**Hint:** you should use your result from problem 1(a) to compute the gradient of the *total function* `loss` $\circ\ (X(\cdot) - y)$ in terms of X, X^T, y, `g`, and x.)

(b) Explain, mathematically, the matrix X, the outputs y, the solution x, the constraint set C, and the outcome of the code.

(c) Run the code multiple times (it's random so the outcome should be different each time). Which model consistently obtains the best fit (judge visually)? Include a plot in which this model obtains the best fit.