Convex Optimization and Modeling	Jun.-Prof. Matthias Hein
Solution of Exercise Sheet 9	16.06.2010

Exercise 19 - Subgradient Steepest Descent

a. **(3 Points)** Consider the minimization of the two-dimensional function

$$f(x_1, x_2) = \begin{cases} 5\sqrt{9x_1^2 + 16x_2^2}, & \text{if } x_1 > |x_2|, \\ 9x_1 + 16|x_2|, & \text{if } x_1 \leq |x_2| \end{cases}$$

using the steepest descent method, which moves from the current point in the opposite direction of the minimum norm subgradient with exact line search. Suppose that the algorithm starts anywhere within the set

$$\{(x_1, x_2) \,|\, x_1 > |x_2| > (9/16)^2 \, |x_1|\}.$$

Verify computationally that it converges to the nonoptimal point $(0, 0)$. What happens if a subgradient method with a constant stepsize is used instead? Check computationally. Provide for both cases the code you have used.

Solution: `function Exercise9`

```
xcur = [2;1];
alpha=0.001;

path=[xcur];
fcur = f(xcur);
fvals = [fcur];

for i=1:50
   subg = gradf(xcur);
   %alpha = ExactLineSearch(xcur,fcur,subg);
   alpha=0.005;
   xnew = xcur - alpha*subg;
   xcur = xnew;
   fvals = [fvals, f(xcur)];
   path=[path, xnew];
end

figure, hold on,
[X,Y]=meshgrid(-1:0.05:2,-1:0.05:1);
Z = zeros(size(X,1),size(X,2));
for i=1:size(X,1),
   for j=1:size(X,2)
      Z(i,j)=f([X(i,j);Y(i,j)]);
   end
end
surfc(X,Y,Z);
numP = size(path,2);
for i=1:numP
```

```
   plot3(path(1,i),path(2,i),fvals(i),'.-','Color',[i/numP,i/numP,i/numP],'MarkerSize',20);
end
hold off


function alpha = ExactLineSearch(xcur,fcur,subg)
% naive exact line search by bisection

% first we search for a point along the line which has larger objective
   % than the current point
   alpha=1;
   NOTFOUND=1;
   while NOTFOUND
      xnew = xcur - alpha*subg;
      fnew = f(xnew);
      if(fnew>fcur)
      NOTFOUND=0;
   else
      alpha=2*alpha;
      if(alpha>1E15)
      error(['Problem seems to be unbounded from below']);
   end
end
end

left=0; right=alpha;
fleft=fcur; fright=f(xcur-alpha*subg);
while (right-left)>1E-15
xnew = xcur - 0.5*(right+left)*subg;
if(-subg'*gradf(xnew)>0) % ascent at the middle point
right=0.5*(right+left);
else % descent at the middle point
left =0.5*(right+left);
end
end
alpha=right;



function fval = f(x) % returns the function value at x
if(x(1)>abs(x(2)))
fval = 5*sqrt(9*x(1)^2 + 16*x(2)^2);
else
fval = 9*x(1) + 16*abs(x(2));
end

function gradfval = gradf(x) % returns the (sub)-gradient of the function at x
if(x(1)>abs(x(2)))
gradfval = 5/2/sqrt(9*x(1)^2 + 16*x(2)^2)*[18*x(1); 32*x(2)];
else
% for the point (0,0) we use the subgradient with minimal norm as
   % required in the steepest descent subgradient method
end
gradfval = 9*[1; 0] + 16*u;
end
```

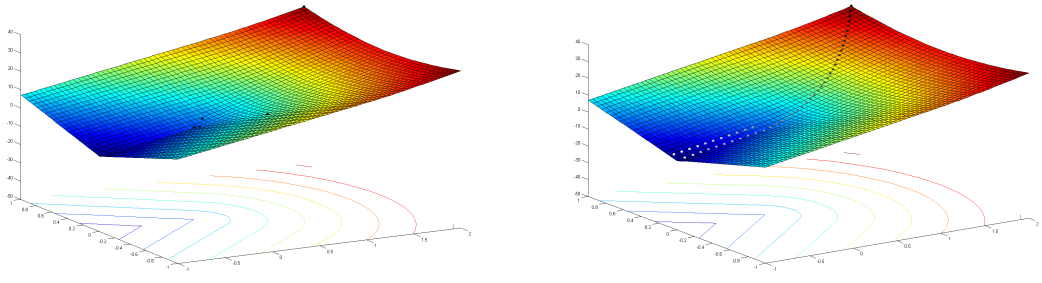

Figure 1: **Left:** Path found by the subgradient method with exact line search. It converges to the origin (if one stops early enough as due to numerical precision the origin is not stable and so the method will eventually converge to $-\infty$ but very, very slow !). The problem is the change of the function from the set where it is differentiable to the non-differentiable set at the origin - where the path x^k has to pass through if one starts in the described area. **Right:** With fixed stepsize the problem does not occur as one simply "jumps" over the difficult place. The oscillating behavior comes from the "jumping" derivative and the fixed stepsize.

Exercise 20 - L_1-Minimization

We want to find the solution of a problem of form,

$$\min_{x \in \mathbb{R}^d} F(x) := f(x) + \lambda \|x\|_1 ,$$

where $\lambda \geq 0$ and f is a convex, continuously twice-differentiable function which has Lipschitz-continuous gradient, that is,

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| ,$$

where L is the Lipschitz-constant. Problems of this type arise in machine learning (loss function + l1-regularization e.g. the lasso), signal processing and image processing.

a. **(3 Points)** Show that the Lipschitz-condition implies that,

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 .$$

b. **(3 Points)** Using the result of the previous part, we derive the following upper bound on the objective $F(x)$. For any $z \in \mathbb{R}^d$,

$$F(x) \leq f(z) + \langle \nabla f(z), x - z \rangle + \frac{L}{2} \|z - x\|^2 + \lambda \|x\|_1 .$$

Derive the closed-form solution of the following problem, which corresponds to the minimization of the upper bound for a particular point z,

$$\arg\min_x \ \langle \nabla f(z), x - z \rangle + \frac{L}{2} \|z - x\|^2 + \lambda \|x\|_1$$

c. **(3 Points)** Implement the following iterative method,

$$x^{(k+1)} = \arg\min_x \ \left\langle \nabla f(x^k), x - x^k \right\rangle + \frac{L}{2} \left\|x^k - x\right\|^2 + \lambda \|x\|_1 ,$$

which minimizes the upper bound on the objective using the point from the previous iteration for the function. This method is called: **Iterative Shrinkage Thresholding Algorithm (ISTA)**. Use the particular quadratic function f,

$$f(x) = \|Ax - Y\|^2,$$

This corresponds to the Lasso-Problem. Data and dummy code will be provided on the webpage.

Solution:

a. The fundamental theorem of differential and integral calculus is as follows,

$$f(y) - f(x) = \int_x^y f'(t)\, dt.$$

The transfer to the multivariate setting can be done by considering the curve, $\gamma(t) = x + t(y - x)$ with tangent vector $\dot{\gamma} = y - x$. The directional derivative of $f(\gamma(t))$ is given as

$$f'(\gamma(t)) = \langle \nabla f(\gamma(t)), \dot{\gamma}(t) \rangle = \langle \nabla f(x + t(y - x)), y - x \rangle .$$

Note, that $f(\gamma(0)) = x$ and $f(\gamma(1)) = y$. Bringing everything together, we get

$$\begin{aligned} f(y) - f(x) - \langle \nabla f(x), y - x \rangle &= \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle \ dt \\ &\leq \int_0^1 \|\nabla f(x + t(y - x)) - \nabla f(x)\| \, \|y - x\| \, dt \\ &\leq \int_0^1 L\, t \ \|y - x\|^2 \ dt \\ &= \frac{L}{2} \|y - x\|^2 . \end{aligned}$$

Thus the Lipschitz-condition for the gradient implies $\|Hf\| \leq L$.

b. We get the optimality condition,

$$0 \in \nabla f(z) + L(x - z) + \lambda u,$$

where $u_i = \text{sign}(x_i)$ if $x_i \neq 0$ and $u_i \in [-1, 1]$ if $x_i = 0$. Reformulating in terms of x,

$$x_i \in z_i - \frac{1}{L}(\nabla f)_i - \frac{\lambda}{L} u_i.$$

The argumentation is now in the same way as done in the lecture. If $z_i - \frac{1}{L}(\nabla f)_i > \frac{\lambda}{L}$, then x_i has to be positive and $x_i = z_i - \frac{1}{L}(\nabla f)_i - \frac{\lambda}{L}$, and analogously if $z_i - \frac{1}{L}(\nabla f)_i < -\frac{\lambda}{L}$, then x_i has to be negative and $x_i = z_i - \frac{1}{L}(\nabla f)_i + \frac{\lambda}{L}$. If $z_i - \frac{1}{L}(\nabla f)_i > \frac{\lambda}{L} \leq \frac{\lambda}{L}$ the only way the equation can be fulfilled, is $x_i = 0$. In total we get,

$$x = S_{\frac{\lambda}{L}}\big(z - \frac{1}{L}\nabla f\big),$$

where S_α is the so-called soft thresholding operator defined as,

$$(S_\alpha(x))_i = \begin{cases} x_i - \alpha, & \text{if } x_i > \alpha, \\ 0, & \text{if } |x_i| \leq \alpha, \\ x_i + \alpha, & \text{if } x_i < -\alpha. \end{cases}$$

c. We first compute the Lipschitz constant of f,

$$\nabla f(x) = 2A^T(Ax - Y),$$

and thus $\nabla f(x) - \nabla f(y) = 2A^T A(x - y)$, from which we deduce $L \leq 2 \left\| A^T A \right\| = 2\lambda_{\max}(A^T A)$.

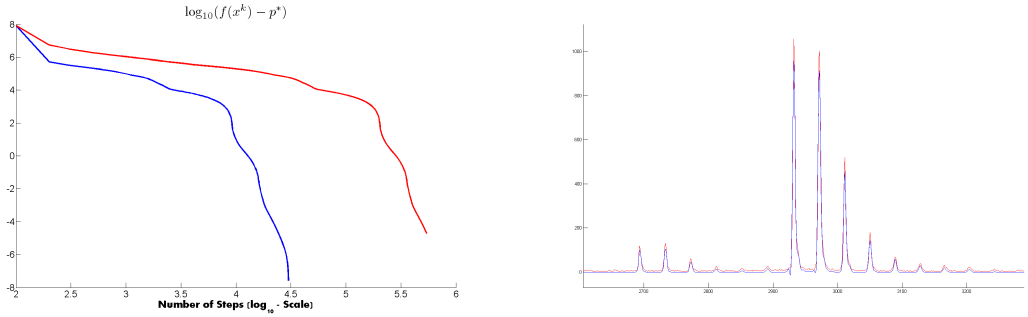

Figure 2: We use as stopping criterion $\frac{f(x^{k+1})-f(x^k)}{f(x^k)} \leq 10^{-14}$. **Left:** The blue line shows the descent for the correct (tight) Lipschitz constant - convergence is achieved in 30000 steps. The red line shows the descent when we replace $\left\|A^T A\right\|$ by $\left\|A^T A\right\|_F$ the Frobenius-norm, which is easy to compute and known to be an upper bound on $\left\|A^T A\right\|$. Convergence then takes 543300 steps - so almost 20 times as long. **Right:** The data Y (red curve) is a mass-spectrogram which consists of a few isotopic patterns, thus apart from the noise the signal has a sparse representation in the "isotopic pattern basis" (plot the columns of A). The solution is sparse (count the non-zeros of x^*) and the fit Ax^* (blue curve) is quite good of without fitting too much the noise. Note, that we have an underfit as we penalize the coefficients.