Convex Optimization and Modeling	Jun.-Prof. Matthias Hein
Solution of Exercise Sheet 11	30.06.2010

Exercise 22 - Projections onto convex sets

a. **(2 Points)** Derive the formula for the projection onto the L_1-ball shown in the lecture.

b. **(2 Points)** Derive the formula for the projection onto the positive semidefinite cone S^n_+ shown in the lecture.

Hint:

- Use the KKT conditions for a).

Solution:

a. The projection as a convex optimization problem is strictly feasible and thus strong duality holds. Thus, necessary and sufficient condition for optimality are,

$$0 \in \nabla \frac{1}{2} \|x - z\|^2 + \lambda \partial \|x\|_1 = x - z + \lambda \text{sign}(x),$$

where $\lambda \geq 0$ is the dual variable of the inequality constraint $\|x\|_1 \leq 1$ and

$$(\text{sign}(x))_i \in \begin{cases} 1 & \text{if } x_i > 0, \\ [-1,1], & \text{if } x_i = 0, \\ -1, & \text{if } x_i < 0. \end{cases}$$

Using the arguments done in the lecture we get

$$x_i = \begin{cases} z_i - \lambda, & \text{if } z_i > \lambda, \\ 0, \text{ if } -\lambda \leq z_i \leq \lambda, & \\ z_i + \lambda, & \text{if } z_i \leq -\lambda. \end{cases}$$

The parameter λ is still undetermined. However, we get from the complementary slackness condition, that $\lambda = 0$ if $\|x\|_1 < 1$ and otherwise $\|x\|_1 = 1$. Using the derived form of x, we obtain,

$$\|x\|_1 = \sum_{i=1}^n \max\{|z_i| - \lambda, 0\} \leq \|z\|_1 .$$

If $\|x\|_1 \leq 1$ we have $\lambda = 0$ and thus $\|x\|_1 = \|z\|_1 \leq 1$. This is the case when the initial point is already inside the L_1-ball, then the projection is just the identity map. In the other case $\|z\|_1 > 1$ and thus the projection x lies at the boundary of the unit-ball and thus $\|x\|_1 = 1$. In this case λ has to be determined by solving,

$$\sum_{i=1}^n \max\{|z_i| - \lambda, 0\} = 1,$$

which can be done using Newton's method. Note that the function is convex in λ.

b. We have to solve the problem for $Z \in S^n$,

$$\min_{X \in S^n_+} \|X - Z\|_F^2 .$$

We transform the problem and go into the eigenbasis of $Z = U\Lambda U^T$ (columns of U), where Λ is the diagonal matrix containing the eigenvalues of Z. Clearly, $Y = U^T X U$ is again positive-semidefinite for any $X \in S_+^n$. Thus with

$$\|X - Z\|_F^2 = \left\|U(U^T X U - \Lambda)U^T\right\|_F^2 = \|Y - \Lambda\|_F^2 ,$$

where we use the fact that the Frobenius norm is invariant under orthogonal transformations which can be seen from,

$$\operatorname{trace}(XZ) = \operatorname{trace}(UYU^T U\Lambda U^T) = \operatorname{trace}(UY\Lambda U^T) = \operatorname{trace}(U^T UY\Lambda) = \operatorname{trace}(Y\Lambda).$$

and $\|A - B\|_F^2 = \operatorname{trace}(A^T A) + \operatorname{trace}(B^T B) - 2\operatorname{trace}(AB)$. we arrive finally at the problem,

$$\min_{Y \in S_+^n} \|Y - \Lambda\|_F^2 .$$

We relax this problem by minimizing over all symmetric matrices with positive diagonal entries. As any positive semi-definite matrix has to have positive diagonal entries, this is a superset of S_+^n.

$$\min_{Y \in S^n,\, Y_{ii} \geq 0, \forall\, i=1,\ldots,n} \|Y - \Lambda\|_F^2 .$$

which has solution $Y_{ii} = \max\{\lambda_i, 0\}$, $i = 1, \ldots, n$ and $Y_{ij} = 0$ for $i \neq j$ (note that the problem can be solved componentwise). Note, that as Y is a diagonal matrix with positive entries it is actually positive semi-definite. Thus the solution of the relaxed problem is positive semi-definite and thus also a solution of the original problem. In the old basis we get $X = \sum_{i=1}^n \max\{\lambda_i, 0\} u_i u_i^T$.

Exercise 23 - Projected Gradient and Subgradient

Implement the projected subgradient and projected gradient method for the non-negative least squares (NNLS) problem.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \frac{1}{2} \|Ax - b\|_2^2 \\ & x \succeq 0. \end{aligned}$$

In both cases use $x^{(0)} = 0$ as starting vector.

a. **(3 Points)** `[xmin,fmin]=ProjectedSubgradientNNLS(MAXITER,A,b)`, where `MAXITER` is the maximal number of steps (there is no stopping criterion - so this is equal to the number of steps). Use a diminishing stepsize $\alpha^k = \frac{c}{k}$. What is a good choice of c ?

b. **(3 Points)** `[xmin,fmin]=ProjectedGradientNNLS(MAXITER,A,b)`. Use $\alpha = 0.1$ and $\beta = \frac{1}{2}$ for the stepsize-selection. Use $\left\| P_C\left(x^{(k)} - \frac{1}{L}\nabla f(x^{(k)})\right) - x^{(k)} \right\| \leq 10^{-10}$ as stopping criterion (together with the upper bound on the number of steps).

c. **(2 Points)** Run both methods with `MAXITER=100000` for the data from Exercise Sheet 9. Plot the logarithm of the relative error, $\log_{10}(\frac{f(x^k)}{p^*} - 1)$, where $p^* = 7.246560956533597e + 005$, for both methods into one plot. Why is the initial solution sparse ? Make the solution x^* even sparser by setting all components of x^* to zero which are smaller than 10. Plot the fit $X * x^*$ versus the mass-spectrogram `Y` in one figure (`hold on` command). What is the effect of the thresholding ?

d. **(2 Points)** The nice property of the projected subgradient method is that one can directly apply it to non-smooth objectives. How would the iterate look like for minimizing

$$\|Ax - b\|_2 + \lambda \|Dx\|_1 ,$$

under the non-negativity constraint (total variation denoising/deblurring with non-negativity constraint) ?

Solution:

a. It turns out that $\alpha^k = \frac{500}{k}$ is a good choice for the stepsize.

Matlab code of `[xmin,fmin]=ProjectedSubgradientNNLS(MAXITER,A,b)`

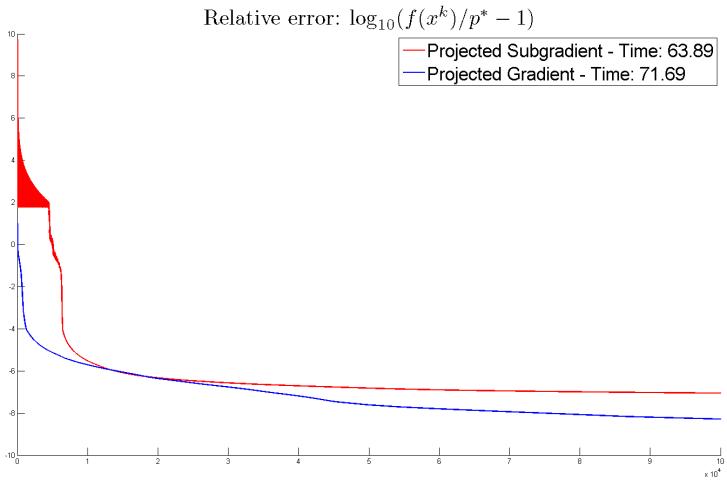
```
function [xmin,fmin,fvals]=ProjectedSubgradientNNLS(MAXITER, A, b)

xcur = zeros(size(A,2),1); % initial point which is feasible
fvals=0.5*norm(b)^2;
fmin=fvals;
xmin=xcur;

tic,
fvals=zeros(MAXITER,1); fminvals=zeros(MAXITER,1); % save all function values

for i=1:MAXITER
alpha=500/i;
res = (A*xcur-b);
subg = A'*res; % (sub)-gradient of objective
xcur = max(0,xcur - alpha*subg); % Projection onto positive orthant
fcur = 0.5*norm(res)^2;
fvals(i)=fcur;
if(fcur<fmin)
fmin=fcur; xmin=xcur;
end
fminvals(i) = fmin;
if(i<100 || rem(i,1000)==0)
disp(['Iteration:  ',num2str(i),' - Current Objective:  ',num2str(norm(fcur,1),'%1.15f')]);
end
end
t=toc,
disp(['Total Time:  ',num2str(t),' - Objective Value:  ',num2str(fmin,'%1.15f')]);
```

Matlab code of `[xmin,fmin]=ProjectedGradientNNLS(MAXITER,A,b)`

```
function [xmin,fmin,fvals]=ProjectedGradientNNLS(MAXITER, A, b)

xcur =zeros(size(A,2),1);
fcur=0.5*norm(A*xcur-b,2)^2;
fmin=fcur;
xmin=xcur;

OPTS.maxit=10;
[u,lambdaMAX] = eigs(sparse(A'*A),1,'LA',OPTS);%sum(sum(BigMatrix.^2));%25.3;
```

```
L = norm(A*u)^2

tic,
runs=MAXITER;
fvals=zeros(runs,1); fminvals=zeros(runs,1);
for i=1:runs
res = (A*xcur-b);
grad = A'*res;
xold = xcur;
xcur = max(0,xcur-1/L*grad);
fold=fcur;
fcur = 0.5*norm(A*xcur-b)^2;
fvals(i)=fcur;
if(fcur<fmin)
fmin=fcur; xmin=xcur;
end
fminvals(i) = fmin;
if(i<10 || rem(i,1000)==0)
disp(['Iteration:  ',num2str(i),' - Current Objective:  ',num2str(norm(fcur,1),'%1.15f')]);
end
if(norm(xcur-xold)<1E-15 || (fold-fcur)<1E-15)
break;
end
end
t=toc,
disp(['Total Time:  ',num2str(t),' - Objective Value:  ',num2str(fmin,'%1.15f')]);
```

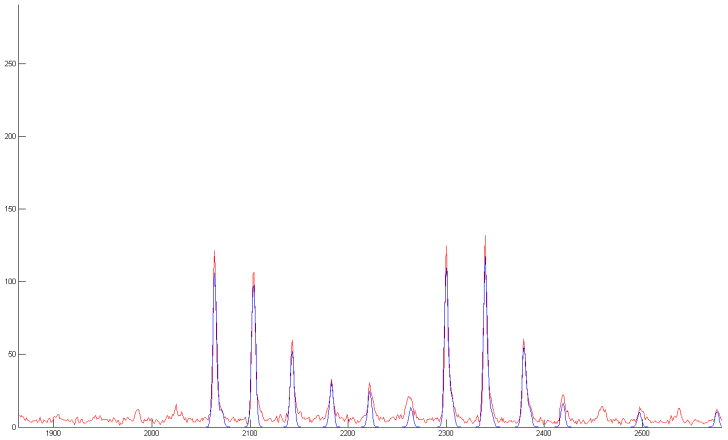
b.



The figure shows the typical behavior of gradient and subgradient method. For both methods we first have a quite quick decay. For the subgradient method it can basically be stopped after 20000 iterations, whereas the gradient method still has some decay. Nevertheless both methods are slow and actually not competitive for this particular dataset. It turns out than an interior point method converges in the fastest way.

The solution x^* is sparse despite we have not included any regularization term penalizing the weights as on sheet 9. The reason is that X is a matrix with positive elements. This

together with the positivity constraint on x results in the sparsity of x as no "positive" and "negative" weights can cancel each other and thus have no effect on the fit. Nevertheless as one just minimizes the fit of the data, also the noise is fitted. However, for this type of data we have a direct interpretation of the coefficient x_i, it is the amplitude of the isotopic peak pattern `X(:,i)`. Thus it makes sense to threshold all small components in order to get the "true" signal (denoising).



d. $x^{k+1} = \max\{x^k - \alpha^k g^k, 0\}$ with

$$g^k = \frac{A^T(Ax-b)}{\|Ax-b\|} + \lambda D^T \text{sign}(Dx),$$

using the chain rule for subgradient, $f(x) = g(Dx)$, $\partial f(x) = D^T \partial g(Dx)$ and taking the derivative of $\|Ax-b\|_2 = \sqrt{\|Ax-b\|_2^2}$.