EE364b Homework 2

1. *Subgradient optimality conditions for nondifferentiable inequality constrained optimization.* Consider the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array}$$

with variable $x \in \mathbf{R}^n$. We *do not* assume that $f_0, \ldots, f_m$ are convex. Suppose that $\tilde{x}$ and $\tilde{\lambda} \succeq 0$ satisfy primal feasibility,

$$f_i(\tilde{x}) \leq 0, \quad i = 1, \ldots, m,$$

dual feasibility,

$$0 \in \partial f_0(\tilde{x}) + \sum_{i=1}^m \tilde{\lambda}_i \partial f_i(\tilde{x}),$$

and the complementarity condition

$$\tilde{\lambda}_i f_i(\tilde{x}) = 0, \quad i = 1, \ldots, m.$$

Show that $\tilde{x}$ is optimal, using only a simple argument, and definition of subgradient. Recall that we do not assume the functions $f_0, \ldots, f_m$ are convex.

Solution. Let g be defined by $g(x) = f_0(x) + \sum_{i=1}^m \tilde{\lambda}_i f_i(x)$. Then, $0 \in \partial g(\tilde{x})$. By definition of subgradient, this means that for any y,

$$g(y) \geq g(\tilde{x}) + 0^T(y - \tilde{x}).$$

Thus, for any y,

$$f_0(y) \geq f_0(\tilde{x}) - \sum_{i=1}^m \tilde{\lambda}_i (f_i(y) - f_i(\tilde{x})).$$

For each i, complementarity implies that either $\lambda_i = 0$ or $f_i(\tilde{x}) = 0$. Hence, for any feasible y (for which $f_i(y) \leq 0$), each $\tilde{\lambda}_i(f_i(y) - f_i(\tilde{x}))$ term is either zero or negative. Therefore, any feasible y also satisfies $f_0(y) \geq f_0(\tilde{x})$, and $\tilde{x}$ is optimal.

2. *Optimality conditions and coordinate-wise descent for ℓ_1-regularized minimization.* We consider the problem of minimizing

$$\phi(x) = f(x) + \lambda \|x\|_1,$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is convex and differentiable, and $\lambda \geq 0$. The number λ is the regularization parameter, and is used to control the trade-off between small f and small $\|x\|_1$. When ℓ_1-regularization is used as a heuristic for finding a sparse x for which $f(x)$ is small, λ controls (roughly) the trade-off between $f(x)$ and the cardinality (number of nonzero elements) of x.

(a) Show that $x = 0$ is optimal for this problem (*i.e.*, minimizes ϕ) if and only if $\|\nabla f(0)\|_\infty \leq \lambda$. In particular, for $\lambda \geq \lambda_{\max} = \|\nabla f(0)\|_\infty$, ℓ_1 regularization yields the sparsest possible x, the zero vector.

Remark. The value $\lambda_{\max}$ gives a good reference point for choosing a value of the penalty parameter λ in ℓ_1-regularized minimization. A common choice is to start with $\lambda = \lambda_{\max}/2$, and then adjust λ to achieve the desired sparsity/fit trade-off.

Solution. A necessary and sufficient condition for optimality of $x = 0$ is that $0 \in \partial\phi(0)$. Now $\partial\phi(0) = \nabla f(0) + \lambda\partial\|0\|_1 = \nabla f(0) + \lambda[-1, 1]^n$. In other words, $x = 0$ is optimal if $-\nabla f(x) \in [-\lambda, \lambda]^n$. This is equivalent to $\|\nabla f(0)\|_\infty \leq \lambda$.

(b) *Coordinate-wise descent.* In the coordinate-wise descent method for minimizing a convex function g, we first minimize over x_1, keeping all other variables fixed; then we minimize over x_2, keeping all other variables fixed, and so on. After minimizing over x_n, we go back to x_1 and repeat the whole process, repeatedly cycling over all n variables.

Show that coordinate-wise descent fails for the function

$$g(x) = |x_1 - x_2| + 0.1(x_1 + x_2).$$

(In particular, verify that the algorithm terminates after one step at the point $(x_2^{(0)}, x_2^{(0)})$, while $\inf_x g(x) = -\infty$.) Thus, coordinate-wise descent need not work, for general convex functions.

Solution. We first minimize over x_1, with x_2 fixed as $x_2^{(0)}$. The optimal choice is $x_1 = x_2^{(0)}$, since the derivative on the left is -0.9, and on the right, it is 1.1. We then arrive at the point $(x_2^{(0)}, x_2^{(0)})$. We now optimize over x_2. But it is optimal, with the same left and right derivatives, so x is unchanged. We're now at a fixed point of the coordinate-descent algorithm.

On the other hand, taking $x = (-t, t)$ and letting $t \to \infty$, we see that $g(x) = -0.1t \to -\infty$.

It's good to visualize coordinate-wise descent for this function, to see why x gets stuck at the crease along $x_1 = x_2$. The graph looks like a folded piece of paper, with the crease along the line $x_1 = x_2$. The bottom of the crease has a small tilt in the direction $(-1, -1)$, so the function is unbounded below. Moving along either axis increases g, so coordinate-wise descent is stuck. But moving in the direction $(-1, -1)$, for example, decreases the function.

(c) Now consider coordinate-wise descent for minimizing the specific function ϕ defined above. Assuming f is strongly convex (say) it can be shown that the iterates converge to a fixed point $\tilde{x}$. Show that $\tilde{x}$ is optimal, *i.e.*, minimizes ϕ.

Thus, coordinate-wise descent works for ℓ_1-regularized minimization of a differentiable function.

Solution. For each i, $\tilde{x}_i$ minimizes the function ψ, with all other variables kept fixed. It follows that

$$0 \in \partial_{x_i}\psi(\tilde{x}) = \frac{\partial f}{\partial x_i}(\tilde{x}) + \lambda \mathcal{I}_i, \quad i = 1, \ldots, n,$$

where $\mathcal{I}_i$ is the subdifferential of $|\cdot|$ at $\tilde{x}_i$: $\mathcal{I}_i = \{-1\}$ if $\tilde{x}_i < 0$, $\mathcal{I}_i = \{+1\}$ if $\tilde{x}_i > 0$, and $\mathcal{I}_i = [-1, 1]$ if $\tilde{x}_i = 0$.

But this is the same as saying $0 \in \nabla f(\tilde{x}) + \partial\|\tilde{x}\|_1$, which means that $\tilde{x}$ minimizes ψ.

The subtlety here lies in the general formula that relates the subdifferential of a function to its partial subdifferentials with respect to its components. For a *separable* function $h : \mathbf{R}^2 \to \mathbf{R}$, we have

$$\partial h(x) = \partial_{x_1} h(x) \times \partial_{x_2} h(x),$$

but this is false in general.

(d) Work out an explicit form for coordinate-wise descent for ℓ_1-regularized least-squares, *i.e.*, for minimizing the function

$$\|Ax - b\|_2^2 + \lambda\|x\|_1.$$

You might find the deadzone function

$$\psi(u) = \begin{cases} u - 1 & u > 1 \\ 0 & |u| \leq 1 \\ u + 1 & u < -1 \end{cases}$$

useful. Generate some data and try out the coordinate-wise descent method. Check the result against the solution found using `CVX`, and produce a graph showing convergence of your coordinate-wise method.

Solution. At each step we choose an index i, and minimize $\|Ax - b\|_2^2 + \lambda\|x\|_1$ over x_i, while holding all other x_j, with $j \neq i$, constant.

Selecting the optimal x_i for this problem is equivalent to selecting the optimal x_i in the problem

$$\text{minimize} \quad ax_i^2 + cx_i + |x_i|,$$

where $a = (A^TA)_{ii}/\lambda$ and $c = (2/\lambda)(\sum_{j \neq i}(A^TA)_{ij}x_j + (b^TA)_i)$. Using the theory discussed above, any minimizer x_i will satisfy $0 \in 2ax_i + c + \partial|x_i|$. Now we note that a is positive, so the minimizer of the above problem will have opposite sign to c. From there we deduce that the (unique) minimizer $x_i^\star$ will be

$$x_i^\star = \begin{cases} 0 & c \in [-1, 1] \\ (1/2a)(-c + \mathbf{sign}(c)) & \text{otherwise,} \end{cases}$$

where

$$\mathbf{sign}(u) = \begin{cases} -1 & u < 0 \\ 0 & u = 0 \\ 1 & u > 0. \end{cases}$$

Finally, we make use of the deadzone function ψ defined above and write

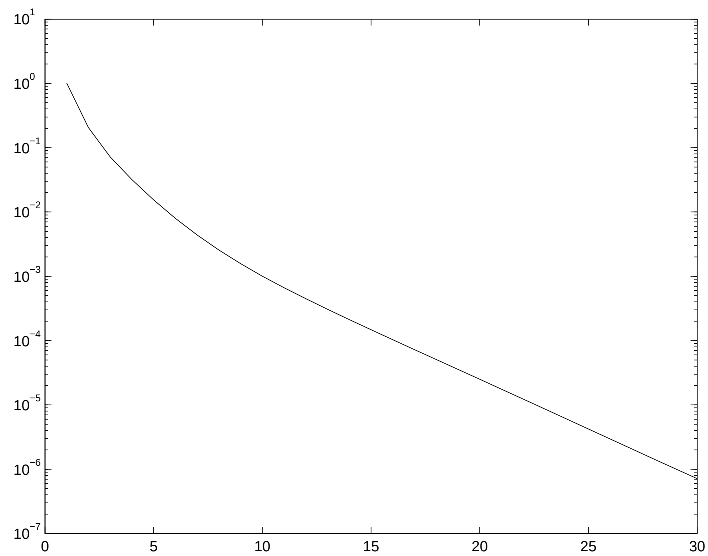
$$x_i^\star = \frac{-\psi((2/\lambda)\sum_{j\neq i}(A^TA)_{ij}x_j + (b^TA)_i)}{(2/\lambda)(A^TA)_{ii}}.$$

Coordinate descent was implemented in Matlab for a random problem instance with $A \in \mathbf{R}^{400\times 200}$. When solving to within 0.1% accuracy, the iterative method required only a third the time of `cvx`. Sample code appears below, followed by a graph showing the coordinate-wise descent method's function value converging to the `CVX` function value.

```
% Generate a random problem instance.
randn('state', 10239); m = 400; n = 200;
A = randn(m, n); ATA = A'*A;
b = randn(m, 1);
l = 0.1;
TOL = 0.001;
xcoord = zeros(n, 1);

% Solve in cvx as a benchmark.
cvx_begin
    variable xcvx(n);
    minimize(sum_square(A*xcvx + b) + l*norm(xcvx, 1));
cvx_end

% Solve using coordinate-wise descent.
while abs(cvx_optval - (sum_square(A*xcoord + b) + ...
                  l*norm(xcoord, 1)))/cvx_optval > TOL
    for i = 1:n
        xcoord(i) = 0; ei = zeros(n,1); ei(i) = 1;
        c = 2/l*ei'*(ATA*xcoord + A'*b);
        xcoord(i) = -sign(c)*pos(abs(c) - 1)/(2*ATA(i,i)/l);
    end
end
```


10^1
10^0
10^−1
10^−2
10^−3
10^−4
10^−5
10^−6
10^−7
0
5
10
15
20
25
30