The University of Texas at Austin
Department of Electrical and Computer Engineering

EE381K: Large Scale Optimization — Fall 2015

Problem Set Four

Constantine Caramanis Due: Thursday, October 1, 2015.

Reading Assignments

1. Reading: Boyd & Vandenberghe: Chapters 9.1 - 9.6.

Matlab and Computational Assignments. Please provide a printout of the Matlab code you wrote to generate the solutions to the problems below.

1. Consider the following univariate function:

$$f(x) = \frac{x}{\sqrt{1+x^2}}.$$

Run un-damped (i.e., step-size = 1) Newton's method, first starting from $x_0 = 3/4$, and then $x_0 = 4/3$, and plot the results. How fast is the convergence in the first case (where $x_0 = 3/4$)?

2. Background: Sequential decision-making tasks are often modeled as a Markov Decision Processes (MDPs). An MDP is defined by the following elements: (1) a state space S, (2) an action space A, (3) next-state transition probabilities conditioned on a state and action $p(s'|s,a)$, and (4) a cost function $c(s,a)$. For this problem, we assume that the state and action spaces are finite. The goal is typically to find a policy $\pi : S \to A$ which minimizes the expected discounted cost:

$$L(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t c(s_t, \pi(s_t))],$$

where $\gamma \in (0,1)$ is a discount factor which weights costs in the near future more heavily. Temporal discounting is one way of modeling infinite horizon problems without $L(\pi)$ diverging.

Rather than encoding the policy directly, a common approach is to define a *value function* $v_\pi(s)$ and have the policy behave greedily with respect to this value function.

$$v_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, \pi(s_t)) | s_0 = s\right].$$

If the value function $v^*(s)$ of the optimal policy π^* can be found, then the optimal actions can be chosen greedily with respect to this value function:

$$\pi^*(s) = \arg\min_{a \in A} \left(c(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v^*(s') \right).$$

One approach to finding v^* formulates the problem as the following linear program (LP)

$$\begin{aligned} \min_{v} \quad & -w^T v \\ \text{s.t.} \quad & c(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v(s') \geq v(s), \ \forall (s,a) \in S \times A, \end{aligned}$$

where w is the vector of (nonnegative) state-relevance weights. To express the problem more compactly, let $P \in \mathbb{R}^{|A||S| \times |S|}$ denote the matrix of transition probabilities where $P_{ij} = p(s_j|(s,a)_i)$, and $M \in \mathbb{R}^{|A||S| \times |S|}$ be the block diagonal matrix extending the $|S|$ dimensional vector v over the $|S||A|$ dimensional joint state-action space.

$$M_{ij} = \begin{cases} 1 & \text{if } i \in [(j-1)|A|, j|A|] \\ 0 & \text{otherwise} \end{cases}$$

Given these definitions, we can restate the LP with the constraints in matrix form:

$$\begin{aligned} \min_{v} \quad & -w^T v \\ \text{s.t.} \quad & (M - \gamma P) v \leq c \end{aligned}$$

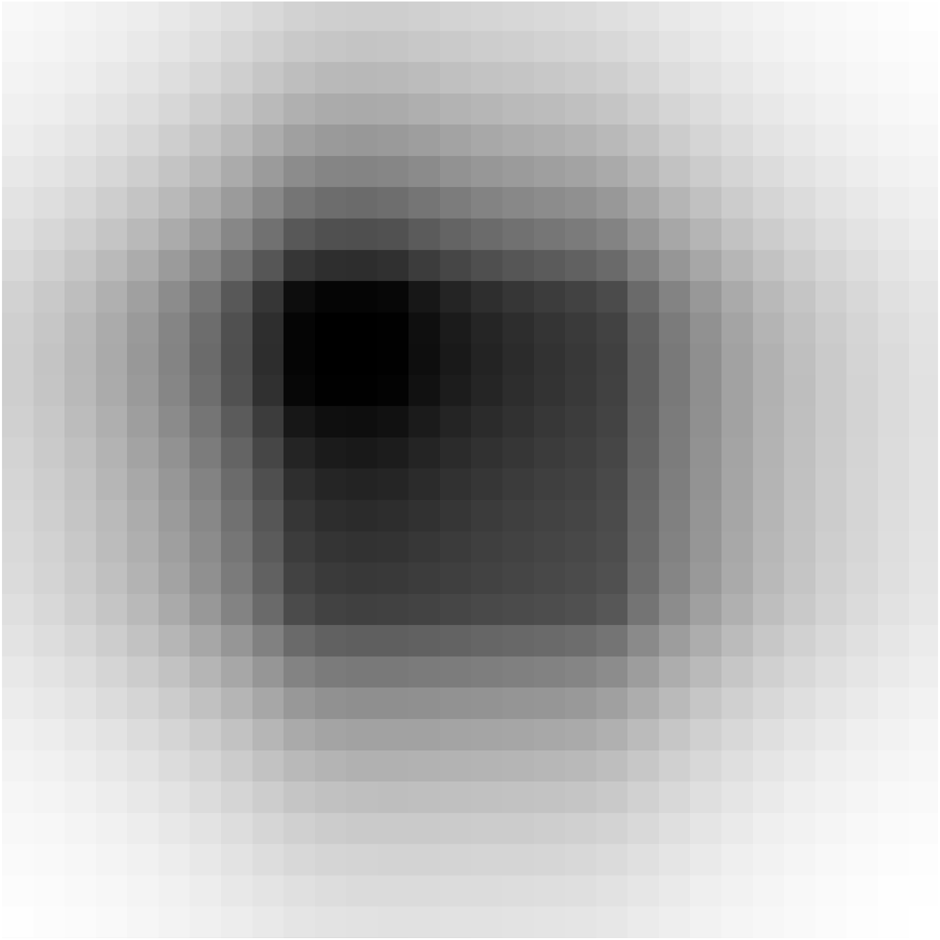
Grid World: We will examine a simple MDP with states arranged as an $n \times n$ grid and four directional actions {U,D,L,R}. The cost function rewards the decision-making “agent” for being close to the center, and the state-action transitions move the agent in the direction of the action chosen 60% of the time and in a uniform random direction 40% of the time.

Using the parameters (γ, w, c, P, M) provided in the zip file, you will solve the LP above using the interior point method discussed in class. Specifically: convert the constrained optimization into an unconstrained problem using the log barrier function. In class, we discussed the formulation where the increasing multiplier t multiplies the linear term. Equivalently, we could have a decreasing multiple, multiplying the log barrier. With this latter formulation, we have a weighting μ (instead of t) multiplying the log barrier, and we can solve this unconstrained problem using Newton's method. Letting A_i denote the i^{th} row of the matrix A, the problem becomes

$$\min_{v} : \ -w^T v - \mu_k \sum_{i} \log(c_i + \gamma P_i v - M_i v).$$

Start with a large enough value for μ_0, and then at each step, (instead of increasing t by a constant factor) decrease μ by a constant factor: $\mu_k = \theta \mu_{k-1}$, $\theta \in (0,1)$. After μ is updated, we solve the new optimization problem via Newton's method, initializing v using the optimal value found by Newton's method from the previous iteration. Halt this process when μ is sufficiently small such that the solution is near the constraint boundary. Plot the final value function over the state grid; does it look correct given the grid world setup described above? It should look something like this:

Figure 1: The optimal grid world value function



Written Problems

1. (?) **Proof of Newton's Method**

   In class we gave part of the proof for Newton's method. We assumed there, that the function to be optimized is strongly convex, with global strong convexity parameter m. Read Section 1.2.4 in Nesterov's book, and see a very similar proof of the quadratic phase of Newton's method that only requires that the function f be strongly convex at the optimizer.

2. **Affine Invariance of Newton's Method**

   Let $f(x)$ be a strongly convex function. Let A be some invertible matrix, and consider the change of coordinates $Ay = x$, and accordingly, the function

$$\phi(y) = f(Ay).$$

   (a) Consider some starting point x_0, and the sequence $\{x_1, x_2, \dots\}$ produced by performing gradient descent on $f(x)$ starting from x_0, and using step-size η_k at iteration k. Define also y_0 given by $Ay_0 = x_0$, and consider the sequence $\{y_1, y_2, \dots\}$ produced by performing gradient descent on $\phi(y)$ starting from y_0, and with the same step size. In general, these will not be the same, i.e., we will not have $Ay_k = x_k$ for all k. Show this, by providing a specific example of a function $f(\cdot)$ and a matrix A. This shows that gradient descent *is not affine invariant.*

(b) Now repeat this for Newton, where the updates $\{x_k\}$ and $\{y_k\}$ are generated by using *undamped* Newton's method on $f(x)$ and $g(y)$, respectively. Show that $Ay_k = x_k$ for all k. This shows that a change of coordinates does not affect what Newton's algorithm is doing.

3. **Gradient descent and non-convexity**

Consider the gradient descent algorithm with fixed step size η for the function $f(x) = x'Qx$, where Q is symmetric but not positive semidefinite. (i.e., Q has some negative eigenvalues). Exactly describe the set of initial points from which gradient descent, with *any* positive step size, will diverge. What happens at the other points, if the step size is small enough ?

4. **Jacobi Method**

Recall that coordinate descent (with exact line search) involves minimizing over one coordinate at a time, keeping the other coordinates fixed. The Jacobi method involves, in a sense, doing all minimizations simultaneously. In particular, given a point x, define the vector $\bar{x}$, in which the value at every coordinate i is determined by the corresponding *individual* coordinate descent update

$$\bar{x}_i \;:=\; \arg\min_{\psi}\, f(x_1, \ldots, x_{i-1}, \psi, x_{i+1}, \ldots, x_n)$$

Thus, potentially, every coordinate of $\bar{x}$ could be different from that of x.

The Jacobi method is defined by the iteration

$$x_+ \;=\; x + \alpha(\bar{x} - x)$$

Prove that, for a convex continuously differentiable f, and a step size $\alpha = 1/n$ where n is the number of coordinates, the next iterate of the Jacobi method produces a lower function value than x, provided x does not already minimize the function.
(Hint: express x_+ as a convex combination of n points.)

5. **Step size in Newton**

Consider the use of Newton's method with constant step size t to minimize the function $\|x\|^3$.

(a) For what values of t do we obtain global convergence to the minimum (i.e., $x^* = 0$) ? What happens for the other values of t ?

(b) For the values of t for which it does converge, why is the convergence not quadratic ?

6. **Composite functions**

Let $f(x) : \mathbb{R}^n \to \mathbb{R}$ be a convex function, $\phi : \mathbb{R} \to \mathbb{R}$ be both convex and increasing, and define $g(x) = \phi(f(x))$. Assume that both f and g are twice differentiable everywhere. Note that this means g too is convex. Note that because $\phi(\cdot)$ is convex and increasing, $g(x)$ is minimized the point x^* that minimizes $f(x)$. Thus one can obtain the solution either by minimizing $g(x) = \phi(f(x))$, or by minimizing $f(x)$. In this problem, we investigate whether a clever choice of $\phi(\cdot)$ could help and provide faster convergence.

(a) Consider an initial point $x^{(0)}$, and run two versions of gradient descent, each with exact line search iterations: one on f, and the other one on g, both starting from the same initial point. Show that the entire sequence of iterates will then be the same, regardless of the choice of $\phi(\cdot)$.

(b) Is the same true for the Newton methods with exact line search ? Prove your answer, or provide a counter-example. *(Hint: use the matrix inversion lemma of Appendix C.4.3 in the book by Boyd and Vandenberghe).*