Notes for 2016-04-22

Computing with constraints

Last time, we discussed different ways of thinking about solving constrained optimization problems of the form

$$\text{minimize } \phi(x) \text{ s.t. } x \in \Omega$$

where the feasible set Ω is defined by equality and inequality conditions

$$\Omega = \{x \in \mathbb{R}^n : c_i(x) = 0, i \in \mathcal{E} \text{ and } c_i(x) \leq 0, i \in \mathcal{I}\}.$$

We will suppose ϕ and all the functions c are differentiable.

Last time, we discussed three approaches to re-casting constrained optimization problems as problems we have already addressed (unconstrained optimization and nonlinear equation solving). The approaches were

- *Constraint elimination*: Find a parameterization $g : \mathbb{R}^m \to \Omega$. This is most straightforward for linear equality constraints.
- *Barriers and penalties*: Add a term to the objective function, dependent on some parameter μ. As $\mu \to 0$, the unconstrained minima of the modified problems converge to the constrained minima of the original/
- *Lagrange multipliers*: Add new variables (multipliers) corresponding to "forces" needed to enforce the constraints[1]. The *KKT conditions* are a set of nonlinear equations in the original unknowns and the multipliers that characterize constrained stationary points.

While we talked about formulations, we spoke hardly at all about algorithms. While there is too little time to do the area any real justice, we will take today to develop one or two ideas in particular contexts.

[1]There are other, more mathematical ways of thinking about Lagrange multipliers. I like thinking of forces because it reduces the chances that I will make a sign error.

Quadratic programs with equality constraints

We begin with a simple case of a quadratic objective and linear equality constraints:

$$\begin{aligned}\phi(x) &= \frac{1}{2}x^T H x - x^T d \\ c(x) &= A^T x - b = 0,\end{aligned}$$

where $H \in \mathbb{R}^{n \times n}$ is symmetric and positive definite, $A \in \mathbb{R}^{n \times m}$ is full rank with $m < n$, and $b \in \mathbb{R}^m$. Not only are such problems useful in their own right, solvers for these problems are also helpful building blocks for more sophisticated problems — just as minimizing an unconstrained quadratic can be seen as the starting point for Newton's method for unconstrained optimization.

Constraint elimination (linear constraints)

As discussed last time, we can write the space of solutions to the constraint equations in terms of a (non-economy) QR decomposition of A:

$$A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

where Q_2 is a basis for the null space of A. The set of solutions satisfying the constraints $Ax = b$ is

$$\Omega = \{u + Q_2 y : y \in \mathbb{R}^{(n-m)}, u = Q_1 R_1^{-T} b\};$$

here u is a *particular solution* to the problem. If we substitute this parameterization of Ω into the objective, we have the unconstrained problem

$$\text{minimize } \phi(u + Q_2 y).$$

While we can substitute directly to get a quadratic objective in terms of y, it is easier (and a good exercise in remembering the chain rule) to compute the stationary equations

$$\begin{aligned}0 = \nabla_y \phi(u + Q_2 y) &= \left(\frac{\partial x}{\partial y}\right)^T \nabla_x \phi(u + Q_2 y) \\ &= Q_2^T (H(Q_2 y + u) - d) = (Q_2^T H Q_2) y - Q_2^T (d - Hu).\end{aligned}$$

In general, even if A is sparse, Q_2 may be dense, and so even if H is dense, we find that $Q_2^T H Q_2$ is dense.

Barriers, penalties, and conditioning

Now consider a penalty formulation of the same equality-constrained optimization function, where the penalty is quadratic:

$$\text{minimize } \phi(x) + \frac{1}{2\mu}\|A^T x - b\|^2.$$

In fact, the augmented objective function is again quadratic, and the critical point equations are

$$(H + \mu^{-1} A A^T) x = d + \mu^{-1} A b.$$

We can analyze this more readily by changing to the Q basis from the QR decomposition of A that we saw in the constraint elimination approach:

$$\begin{bmatrix} Q_1^T H Q_1 + \mu^{-1} R_1 R_1^T & Q_1^T H Q_2 \\ Q_2^T H Q_1 & Q_2^T H Q_2 \end{bmatrix} (Q^T x) = \begin{bmatrix} Q_1^T d + \mu^{-1} R_1 b \\ Q_2^T d \end{bmatrix}$$

Taking a Schur complement, we have

$$(\mu^{-1} R_1 R_1^T + F)(Q_1^T x) = \mu^{-1} R_1 b - g$$

where

$$\begin{aligned} F &= Q_1^T H Q_1 - Q_1^T H Q_2 (Q_2^T H Q_2)^{-1} Q_2^T H Q_1 \\ g &= [I - Q_1^T H Q_2 (Q_2^T H Q_2)^{-1} Q_2^T] d \end{aligned}$$

As $\mu \rightarrow 0$, the first row of equations is dominated by the μ^{-1} terms, and we are left with

$$R_1 R_1^T (Q_1^T x) - R_1 b \rightarrow 0$$

i.e. $Q_1 Q_1^T x$ is converging to $u = Q_1 R_1^{-T} b$, the particular solution that we saw in the case of constraint elimination. Plugging this behavior into the second equation gives

$$(Q_2^T H Q_2)(Q_2^T x) - Q_2^T (d - Hu) \rightarrow 0,$$

i.e. $Q_2^T x$ asymptotically behaves like y in the previous example. We need large μ to get good results if the constraints are ill-posed or if $Q_2^T H Q_2$ is close to singular. But in general the condition number scales like $O(\mu^{-1})$, and so large values of μ correspond to problems that are numerically unattractive.

Lagrange multipliers and KKT systems

The KKT conditions for our equality-constrained problem say that the gradient of

$$L(x, \lambda) = \phi(x) + \lambda^T (A^T x - b)$$

should be zero. In matrix form, the KKT system (saddle point system)

$$\begin{bmatrix} H & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} d \\ b \end{bmatrix}.$$

If A and H are well-conditioned, then so is this system, so there is no bad numerical behavior. The system also retains whatever sparsity was present in the original system matrices H and A. However, adding the Lagrange multipliers not only increases the number of variables, but the extended system lacks any positive definiteness that H may have.

The KKT system is closely related to the penalty formulation that we saw in the previous subsection, in that if we use Gaussian elimination to remove the variable λ in

$$\begin{bmatrix} H & A \\ A^T & -\mu I \end{bmatrix} \begin{bmatrix} \hat{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} d \\ b \end{bmatrix},$$

we have the Schur complement system

$$(H + \mu^{-1} A A^T) \hat{x} = d + \mu^{-1} A b,$$

which is identical to the stationary point condition for the quadratically penalized objective.

Other approaches

Of course, the three approaches we have sketched above are not the only methods! For example, augmented Lagrangian methods combine a penalty approach with an approximate Lagrange multiplier, alternately solving an unconstrained problem and updating a set of multiplier estimates. Other methods (e.g. projected gradient descent) are available if there is a cheap way to project a proposed solution to the closest point on the constraint set. And one can apply the same types of inexact iterations and quasi-Newton ideas we saw in the context of unconstrained optimization.

Inequality constraints

Problems with inequality constraints can be reduced to problems with *equality* constraints if we can only figure out which constraints are active at the solution. We use two main strategies to tackle this task:

- *Active set* methods guess which constraints are active, then solve an equality-constrained problem. If the solution satisfies the KKT conditions, we are done. Otherwise, we update the guess of the active set by looking for constraint violations or negative multipliers. The *simplex method* for linear programs is a famous active set method. The difficulty with these methods is that it may take many iterations before we arrive on the correct active set.

- *Interior point* methods take advantage of the fact that barrier formulations do not require prior knowledge of the active constraints; rather, the solutions converge to an appropriate boundary point as one changes the boundary.

Between the two, active set methods often have an edge when it is easy to find a good guess for the constraints. Active set methods are great for families of related problems, because they can be "warm started" with an initial guess for what constraints will be active and for the solution. Many strong modern solvers are based on sequential quadratic programming, a Newton-like method in which the model problems are linearly-constrained quadratic programs that are solved by an active set iteration. In contrast to active set methods, interior point methods spend fewer iterations sorting out which constraints are active, but each iteration may require more work.

Solving least squares on a simplex

A couple years ago, I was faced with a machine learning task in which a key subproblem involved a large number of small simplex-constrained least squares problems:

$$\text{minimize } \frac{1}{2}\|Ax - b\|^2 \text{ s.t. } x \geq 0 \text{ and } \sum_i x_i = 1.$$

The matrix A remained constant across all instances of the problem; only the right hand side b changed. I did not have an appropriate specialized solver at hand, and the general-purpose solvers at my disposal were all a bit heavy-weight for the context where I wanted them. I followed a combined strategy that used several of the ideas we have discussed this week, and wanted to close the lecture with a few of them.

Problem reduction and initial guess

First, recall that if $A = QR$ is an economy QR decomposition, then

$$\|Ax - b\|^2 = \|Rx - Q^T b\|^2 + \|(I - QQ^T)b\|^2.$$

This decomposition of the squared residual norm has nothing to do with how we constraint x. Hence, we can rewrite the simplex-constrained problem as

$$\text{minimize } \frac{1}{2}\|Rx - \tilde{b}\|^2 \text{ s.t. } x \geq 0 \text{ and } \sum_i x_i = 1,$$

where $\tilde{b} \equiv Q^T b$.

As an initial guess, we project the solution to the unconstrained problem onto the simplex (i.e. the set Ω of points with non-negative coordinates that sum to one). I found how to do this via Google; it's in Figure 1 in an ICML 2008 paper by Duchi *et al.* With this initial guess, we have a proposed active set for the first step of the algorithm.

Normalization constraint

Suppose $\mathcal{J}$ is the set of variables for which the non-negativity constraint is inactive. That is, we guess that a solution will satisfy

$$\begin{cases} x_j > 0, & j \in \mathcal{J} \\ x_j = 0, & j \notin \mathcal{J} \end{cases}$$

We will assume (without loss of generality, as we shall see) that $\mathcal{J} = \{1, 2, \ldots, \|\mathcal{J}|\}$; in this case, we want to solve the equality-constrained subproblem for a step from the current solution estimate $x_{\mathcal{J}}$ to the next one:

$$\text{minimize } \frac{1}{2}\|\bar{R}p - f\|^2 \text{ s.t. } \sum_{j \in \mathcal{J}} p_j = 0,$$

where $\bar{R} = R(\mathcal{J}, \mathcal{J})$ is a leading submatrix of R and $f = b(\mathcal{J}) - \bar{R}x_{\mathcal{J}}$ is the leading subvector of the current residual. Were it not for the equality constraint, we could solve this by back-substitution. To enforce the equality constraint, we add a Lagrange multiplier, leading to the KKT system

$$\begin{bmatrix} \bar{R}^T\bar{R} & e \\ e^T & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} \bar{R}^T f \\ 0 \end{bmatrix}.$$

We can multiply the first block row by $\bar{R}^{-T}$ and the first block column by $\bar{R}^{-1}$ to find

$$\begin{bmatrix} I & w \\ w^T & 0 \end{bmatrix} \begin{bmatrix} \bar{R}p \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

where $w \equiv \bar{R}^{-T}e$. Gaussian elimination on this system gives

$$\begin{aligned} \lambda &= \frac{w^T f}{w^T w} \\ \bar{R}x_{\mathcal{J}} &= f - \lambda w \end{aligned}$$

Hence, solving the constrained system involves one additional triangular solve (to compute w) and a few dot products beyond the usual triangular solve we would use for the unconstrained problem.

Once we have a direction, we take a step in that direction. If taking a full step violates a constraint, we take the longest partial step that does not violate any constraints, and we add the constraint that controls the step size to the proposed active set. Otherwise, after taking a full step, we check whether all the constraints marked as active really should be active. If one or more constraints should not be active, we remove the one that is "worst" (in the sense of the most negative Lagrange multiplier) from the active set.

Back to numerical linear algebra

We like the convention that the first columns of $\bar{R}$ are correspond to the active elements. To maintain this convention, we need primitives to update the R factor in a QR factorization after transposing two columns of the original matrix. That is, we want to be able to

- Start with a triangular matrix R
- Swap columns j and $j' > j$ to get a new matrix $\tilde{R}$

- Apply orthogonal transformations from the right to get a new upper triangular matrix R'.

Because this happens at every step of the nonnegative least squares iteration, we want it to run fast. We do this in the follosing steps:

- $\tilde{R}$ is not triangular because there are too many nonzeros in column j. To eliminate these nonzeros, we make an "upward pass" where we apply a sequence of Givens rotations on row pairs $(j'-1, j'), (j'-2, j'-1), \ldots, (j+1, j+2)$ to introduce zeros in column j below the first subdiagonal.

- After applying the first set of Givens rotations, our matrix is now in upper Hessenberg form. To restore to triangular form, we make a "downward pass" where we use Givens rotations on row pairs $(j, j+1)$ through $(j'-1, j')$ to zero out the subdiagonal entries in columns j through j'.

The cost of applying a Givens rotation across the rows of the matrix is $O(n)$, so the total cost of this two-sweep update is $O(n(j'-j))$.

The Givens-based matrix update/downdate technology lets us compute each step in the active set iteration in at most $O(n^2)$ time — and potentially less. Because we are using the same R matrix for many distinct problems, we do not even require an $O(n^3)$ factorization for each new right hand side! In contrast, a solver that used barriers or penalties would usually require $O(n^3)$ time per iteration. Consequently, a little care in our choice of algorithms — and in our linear algebra implementations — can lead to a big performance boost.