Notes for 2016-04-27

Seeking structure

For the past three weeks, we have discussed rather general-purpose optimization methods for nonlinear equation solving and optimization. In practice, of course, we should look at our problems to see if they have structure we can use in specialized algorithms that are faster or more robust than the general-purpose methods. At one end of the spectrum, there are specialized solvers for specific problem types, such as linear programming problems, linear least-squares problems with ℓ^1 regularizers (LASSO), or non-negative least squares problems. More generally, there are less-specialized solvers (and proof techniques) for more general classes such as convex optimization problems. This week, we focus on two classes of structured nonlinear equation solvers and optimization methods with deep connections to linear algebraic ideas from the start of the semester.

Eigenvalue problems and optimization

There is a sharp division in the optimization world between *convex* problems and *nonconvex* problems. The former are tractable; the latter are often very hard indeed. The symmetric eigenvalue problem is in a sweet spot between the two: it covers a broad class of interesting non-convex problems, but we know how to solve it efficiently.

NB: I talked about Rayleigh quotients and about graph partitioning in class, but not the other examples.

The marvelous Rayleigh quotient

In our discussion of eigenvalue problems, we have already seen the *Rayleigh quotient*

$$\rho_A(x) = \frac{x^T A x}{x^T x}.$$

We commented earlier in the semester that the stationary points of the Rayleigh quotient are the eigenvectors of A, and the stationary values are the corresponding eigenvalues.

Because $\rho_A(\alpha x) = \rho_A(x)$ for any nonzero α, we can rephrase the problem of finding stationary points of the Rayleigh quotient as finding critical points of $x^T Ax$ constrained to the unit sphere. Using Lagrange multipliers, this is equivalent to finding stationary points (with respect to x and λ) of

$$L(x, \lambda) = x^T Ax - \lambda(x^T x - 1).$$

The gradient of L with respect to x is

$$\nabla_x L = 2(Ax - \lambda x);$$

and so we see that the eigenvalue can also be interpreted as a Lagrange multiplier.

More generally, we can consider minimizing $x^T Ax$ subject to the constraint that $x^T Mx = 1$; the constrained fixed point equations then take the form of a *generalized eigenvalue problem*:

$$Ax = \lambda Mx.$$

If M is symmetric and positive definite, we can convert this generalized problem to a standard eigenvalue problem by a change of variables. However, it is often useful to keep such problems in generalized form, as the conversion to a standard eigenvalue problem may destroy sparsity.

Quadratic constraints

What if we move a step beyond the pure quadratic problem? For example, consider

$$\text{minimize } \frac{1}{2}x^T Ax - b^T x \text{ s.t. } x^T x = 1.$$

The KKT conditions are

$$\begin{aligned}(A - \lambda I)x &= b \\ x^T x &= 1.\end{aligned}$$

This is not itself an eigenvalue problem, but we can get there. Note that

$$b^T(A - \lambda I)^{-2}b - 1 = 0,$$

which, with a little cleverness, we can re-interpret as a zero in the final Schur complement of

$$\begin{bmatrix} (A-\lambda)^2 & b \\ b^T & 1 \end{bmatrix}.$$

Eliminating the last variable instead of the first gives the *quadratic eigenvalue problem*

$$\left[(A-\lambda I)^2 - bb^T\right] u = 0,$$

where $v = (A-\lambda u)$ is proportional to a constrained stationary point. We can also go one step further and reduce to a standard eigenvalue problem:

$$\begin{bmatrix} (A-\lambda I) & -I \\ -bb^T & (A-\lambda I) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 0.$$

This gives us the following characterization of all stationary points of the constrained problem: for each real solution to the eigenvalue problem

$$\begin{bmatrix} A & -I \\ -bb^T & A \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix},$$

the vectors $\pm v/\|v\|$ are stationary points for the constrained problem. In this case, it is faster to deal directly with the quadratic problem; in fact, it is reasonably straightforward to use a divide-and-conquer idea to solve this problem in $O(n^2)$ time after a symmetric eigendecomposition of A. But this is perhaps a topic for a different time.

What if we went one step more complicated? For example, we might try to optimize subject to a constraint $x^T M x - 2d^T x + c = 0$. But if M is symmetric and positive definite with the Cholesky factorization $M = R^T R$, then the change of variables

$$x = R^{-1}(R^{-T}d + z)$$

gives us the constraint equation

$$z^T z = -\left(\|R^{-T}d\|^2 + c\right)$$

and a quadratic objective function. Hence, we can still deal with this case by reduction to an eigenvalue problem. On the other hand, if we have more than one quadratic constraint, things become rather more complicated.

Optimal matrix approximations

A wide variety of *matrix nearness* problems can be converted to eigenvalue problems. Often this is an optimization with a quadratic objective and quadratic constraints in disguise. As a few examples, we mention the nearest low-rank matrix to a given target (computed by the SVD via the Eckart-Young theorem), the nearest positive semi-definite matrix to a given symmetric matrix (typically computed via a symmetric eigenvalue decomposition), the nearest orthogonal matrix to a given matrix (the solution to an *orthogonal Procrustes problem*), and the distance between a matrix and the nearest unstable matrix (computed via the Byers[1]-Boyd-Balikrishnan algorithm). Nick Higham has written a good deal on matrix nearness problems; see, e.g. the survey article "Matrix Nearness Problems and Applications."

Dimensionality reduction

Students often first encounter the SVD when learning about principal components analysis (PCA), one of the basic forms of dimensionality reduction. There are a variety of other dimensionality reduction methods that, like PCA, turn into *trace optimization* problems:

$$\text{optimize } \operatorname{tr}(X^T A X) \text{ s.t. } X^T X = I,$$

where the optimization may be a minimization or maximization depending on the context (and the matrix A). We will spell out this example in a little more detail as a reminder of the value of variational notation. We can write the augmented Lagrantian in this case as

$$L(X, M) = \operatorname{tr}(X^T A X) - \operatorname{tr}(M^T (X^T X - I))$$

where M is now a *matrix* of Lagrange multipliers, one for each of the scalar constraints in the matrix equation $X^T X = I$. Taking variations gives

$$\delta L = 2 \operatorname{tr}(\delta X^T (AX - XM)) + \operatorname{tr}(\delta M^T (X^T X - I)),$$

[1]Ralph Byers was a student of Charlie Van Loan who went on to do really interesting work in numerical linear algebra, and particularly on eigenvalue problems. I met him while I was a graduate student at Berkeley; he was busy putting his aggressive early deflation ideas into the LAPACK nonsymmetric eigensolver. He was a kind and interesting gentleman. He passed away in December 2007, just less than a month after Gene Golub's death. He is still missed.

which gives us the equation

$$AX = XM.$$

for some M. This is an *invariant subspace* equation. The equations allow X is an arbitrary orthonormal basis for the subspace in question (the subspace associated with the largest or smallest eigenvalues depending whether we are interested in trace minimization or maximization).

For a good overview of the connections between trace optimization and dimensionality reduction, we refer to "Trace optimization and eigenproblems in dimension reduction methods" by Kokiopoulou, Chen, and Saad.

Combinatorial connections

We have already seen that we can optimize a quadratic subject to one quadratic constraint in the real case. Integer or binary quadratic programs are harder, but we can often use a *spectral relaxation* to approximately solve these programs. As an example, we describe a standard spectral method for a classic problem in graph theory: bisecting a graph with as few cut edges as possible.

Consider an undirected graph $G = (V, E)$, $E \subset V \times V$. We partition V into disjoint subsets V^+ and V^- by a mapping $u : V \to \{\pm 1\}$. The number of edges going between V^+ and V^- is

$$\frac{1}{4} \sum_{(i,j) \in E} (u_i - u_j)^2 = \frac{1}{2} u^T L u$$

where L is the *graph Laplacian matrix*

$$L_{ij} = \begin{cases} \text{degree of } i, & i = j \\ -1, & i \neq j \text{ and } (i,j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

The difference in the size of V^+ and V^- is $e^T u = \sum_i u_i$. Hence, bisection with a minimum cut corresponds to the binary quadratic program

$$\text{minimize } u^T L u \text{ s.t. } e^T u = 0 \text{ and } u \in \{\pm 1\}^n.$$

The general graph partitioning problem is NP-hard; but the problem is not in the objective or the constraint that $e^T u = 0$, but the constraint that

$u \in \{\pm 1\}^n$. We can *relax* this constraint to the condition that $\sum_i u_i^2 = n$; that is,

$$\text{minimize } u^T L u \text{ s.t. } e^T u = 0 \text{ and } \|u\|^2 = n, u \in \mathbb{R}^n.$$

The vector e is itself a null vector of L; if the graph is connected, all other eigenvalues of L are positive. The solution to the relaxed problem is that u is a scaled eigenvector corresponding to the second-smallest eigenvalue $\lambda_2(L)$; the value of $u^T L u$ is $n\lambda_2(L)$. Because the continuous optimization is over a larger set than the discrete optimization, $n\lambda_2(L)$ is a lower bound on the minimum number of edges needed for bisection. The eigenvalue $\lambda_2(L)$ is sometimes called the *algebraic connectivity* of the graph; when it is large, there are no small cuts that can bisect the graph The corresponding eigenvector is the *Fiedler vector*. Although the entries of the Fiedler vector are no longer ± 1, using the sign of the entries as a method of partitioning often works quite well. This is the basis of *spectral partitioning*.

Eigenvalues and global root finding

So far, we have discussed spectral approaches to optimization problems. But spectral methods are also relevant to nonlinear equation solving, particularly in one variable. The basic picture is:

- If $f : [a, b] \to \mathbb{R}$ is a smooth function, we approximate it by a polynomial p to arbitrary accuracy.
- We re-interpret the problem of finding roots of p as the problem of finding eigenvalues of A s.t. $p(z) \propto \det(zI - A)$.

There is a standard trick to finding a matrix eigenvalue problem corresponding to a polynomial root-finding problem; it is the same trick that's used to convert a differential equation to first-order form. Define a recurrence relation for powers

$$v_j = \lambda^j = \lambda v_{j-1}$$

Then we can rewrite

$$0 = p(\lambda) = \lambda^d + \sum_{j=0}^{d-1} a_j \lambda^j$$

as

$$0 = \lambda v_{d-1} + \sum_{j=0}^{d-1} a_j v_j.$$

Putting this together with the recurrence relation, we have

$$\begin{bmatrix} -a_{d-1} & -a_{d-2} & \dots & -a_1 & -a_0 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix} \begin{bmatrix} v_{d-1} \\ v_{d-2} \\ v_{d-3} \\ \vdots \\ v_0 \end{bmatrix} = \lambda \begin{bmatrix} v_{d-1} \\ v_{d-2} \\ v_{d-3} \\ \vdots \\ v_0 \end{bmatrix}.$$

This matrix with polynomial coefficients across the top is a *companion matrix*. It is a highly structured Hessenberg matrix, although exploiting the structure in a numerically stable way turns out to be nontrivial. The MATLAB `roots` command computes roots of a polynomial by running the ordinary QR eigensolver on such a matrix.

More generally, if p is expressed in terms of a basis that can be evaluated by some recurrence relation (e.g. Chebyshev or Legendre polynomials), then there is an associated Hessenberg *confederate matrix* whose eigenvalues are the roots of the polynomial. For the Chebyshev polynomials, the confederate matrix is sometimes called a *comrade matrix*[2]; there is also something called a *colleague matrix*. The trick is also not restricted to expansions in terms of polynomials; for example, the problem of finding roots of a function that is well approximated by a finite Fourier series can similarly be converted into an eigenvalue problem[3].

Nonlinear eigenvalue problems

A *nonlinear eigenvalue problem* is a problem of the form

$$\text{Find } (\lambda, u) \text{ with } u \neq 0 \text{ s.t. } T(\lambda)u = 0.$$

Your third project is an example of a nonlinear eigenvalue problem (the eigenvalue is the equilibrium resource level r), but there are many other

[2] Товарищ?

[3] See, for example, *Solving Transcendental Equations* by J.P. Boyd. This is a book worth remembering even if you have decided you are only mildly interested in equation solving. Boyd is an unusually entertaining writer.

examples that come from analyzing the stability of linear differential equations and difference equations. We can approximate many nonlinear eigenvalue problems by *polynomial* eigenvalue problems $P(\lambda)u = 0$ where

$$P(z) \approx \sum_{j=0}^{d} z^j A_j.$$

This problem, too, can be encoded as a matrix eigenvalue problem; if A_d is nonsingular, we have

$$\begin{bmatrix} -\bar{A}_{d-1} & -\bar{A}_{d-2} & \dots & -\bar{A}_1 & -\bar{A}_0 \\ I & & & & \\ & I & & & \\ & & \ddots & & \\ & & & I & 0 \end{bmatrix} \begin{bmatrix} v_{d-1} \\ v_{d-2} \\ v_{d-3} \\ \vdots \\ v_0 \end{bmatrix} = \lambda \begin{bmatrix} v_{d-1} \\ v_{d-2} \\ v_{d-3} \\ \vdots \\ v_0 \end{bmatrix},$$

where $\bar{A}_j = A_d^{-1} A_j$.

Spectral summary

A wide variety of problems can be reduced to or approximated by either a quadratic program with one quadratic equality constraint or by a polynomial root-finding problem. When you see a problem with this structure, most likely there is an eigenvalue problem lurking nearby.