2018-06-14

1 Bias-variance tradeoffs and regularization

At the end of the last class, we talked about generalization error for least squares in the presence of noise. In particular, we saw that if there is a true linear model $Ax = b$ and we compute an estimate $A_1\hat{x} = \hat{b}_1$ involving a subset of the data with noisy measurements ($\hat{b}_1 = b_1 + e$), then

$$\|Ax - A\hat{x}\| \leq \|AA_1^\dagger\|\|e\| \leq \|A\|\|A_1^\dagger\|\|e\|.$$

We cannot generally overcome the effects of the measurement error, but we can at least hope not to amplify them. The amplification factor $\|A\|\|A_1^\dagger\|$ could be large if $\|A\|$ is large, but a more common problem is that $\|A_1^\dagger\|$ is large. In numerical analysis, we might call this a problem with *ill-conditioning*; in statistics, we might refer to this as an issue with *high variance* in the estimator.

More generally, suppose we have $y = f(x) + \epsilon$ where ϵ is a noise term with mean zero and variance σ^2, and we use data y to fit an estimator function $\hat{f}(x)$. Then for an unseen point x

$$\mathbb{E}[(y - \hat{f}(x))^2] = \mathbb{E}[\hat{f}(x) - f(x)]^2 + \mathrm{Var}[\hat{f}(x)] + \sigma^2;$$

that is, the squared error involves the squared *bias* (caused by inability of the model to fit the function even with perfect data); the *variance* (associated with stability of the estimation procedure under perturbations from noise); and a term associated with measurement error.

If we have an ill-conditioned fitting problem (or a high variance estimator, if you prefer), what should we do? There are three options:

1. Get more data.
2. Reduce the noise.
3. Change the fitting problem to reduce the ill-conditioning.

Getting more data or reducing the noise is not always practical or economical, and in any event is an issue unrelated to numerical methods. Thus, we will turn now to the last approach: *regularizing* the problem to reduce the ill-conditioning, possibly at the expense of a small bias.

2 Factor selection and pivoted QR

In ill-conditioned problems, the columns of A are nearly linearly dependent; we can effectively predict some columns as linear combinations of other columns. The goal of the column pivoted QR algorithm is to find a set of columns that are "as linearly independent as possible." This is not such a simple task, and so we settle for a greedy strategy: at each step, we select the column that is least well predicted (in the sense of residual norm) by columns already selected. This leads to the *pivoted QR factorization*

$$A\Pi = QR$$

where Π is a permutation and the diagonal entries of R appear in descending order (i.e. $r_{11} \geq r_{22} \geq \ldots$). To decide on how many factors to keep in the factorization, we either automatically take the first k or we dynamically choose to take k factors where r_{kk} is greater than some tolerance and $r_{k+1,k+1}$ is not.

The pivoted QR approach has a few advantages. It yields *parsimonious* models that predict from a subset of the columns of A – that is, we need to measure fewer than n factors to produce an entry of b in a new column. It can also be computed relatively cheaply, even for large matrices that may be sparse.

3 Tikhonov

A second approach is to say that we want a model in which the coefficients are not too large. To accomplish this, we add a penalty term to the usual least squares problem:

$$\text{minimize } \|Ax - b\|^2 + \lambda^2 \|x\|^2.$$

Equivalently, we can write

$$\text{minimize } \left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|^2,$$

which leads to the regularized version of the normal equations

$$(A^T A + \lambda^2 I)x = A^T b.$$

In some cases, we may want to regularize with a more general norm $\|x\|_M^2 = x^T M x$ where M is symmetric and positive definite, which leads to the regularized equations

$$(A^T A + \lambda^2 M)x = A^T b.$$

If we know of no particular problem structure in advance, the standard choice of $M = I$ is a good default.

It is useful to compare the usual least squares solution to the regularized solution via the SVD. If $A = U\Sigma V^T$ is the economy SVD, then

$$x_{LS} = V\Sigma^{-1}U^T b$$
$$x_{Tik} = V f(\Sigma)^{-1} U^T b$$

where

$$f(\sigma)^{-1} = \frac{\sigma}{\sigma^2 + \lambda^2}.$$

This *filter* of the inverse singular values affects the larger singular values only slightly, but damps the effect of very small singular values.

4 Truncated SVD

The Tikhonov filter reduces the effect of small singular values on the solution, but it does not eliminate that effect. By contrast, the *truncated SVD* approach uses the filter

$$f(z) = \begin{cases} z, & z > \sigma_{\min} \\ \infty, & \text{otherwise.} \end{cases}$$

In other words, in the truncated SVD approach, we use

$$x = V_k \Sigma_k^{-1} U_k^T b$$

where U_k and V_k represent the leading k columns of U and V, respectively, while Σ_k is the diagonal matrix consisting of the k largest singular values.

5 ℓ^1 and the lasso

An alternative to Tikhonov regularization (based on a Euclidean norm of the coefficient vector) is an ℓ^1 regularized problem

$$\text{minimize } \|Ax - b\|^2 + \lambda \|x\|_1.$$

This is sometimes known as the "lasso" approach. The ℓ^1 regularized problem has the property that the solutions tend to become sparse as λ becomes larger. That is, the ℓ^1 regularization effectively imposes a factor selection process like that we saw in the pivoted QR approach. Unlike the pivoted QR approach, however, the ℓ^1 regularized solution cannot be computed by one of the standard factorizations of numerical linear algebra.

6 Tradeoffs and tactics

All four of the regularization approaches we have described are used in practice, and each has something to recommend it. The pivoted QR approach is relatively inexpensive, and it results in a model that depends on only a few factors. If taking the measurements to compute a prediction costs money — or even costs storage or bandwidth for the factor data! — such a model may be to our advantage. The Tikhonov approach is likewise inexpensive, and has a nice Bayesian interpretation (though we didn't talk about it). The truncated SVD approach involves the best approximation rank k approximation to the original factor matrix, and can be interpreted as finding the k best factors that are linear combinations of the original measurements. The ℓ_1 approach again produces models with sparse coefficients; but unlike QR with column pivoting, the ℓ_1 regularized solutions incorporate information about the vector b along with the matrix A.

So which regularization approach should one use? In terms of prediction quality, all can provide a reasonable deterrent against ill-posedness and overfitting due to highly correlated factors. Also, all of the methods described have a parameter (the number of retained factors, or a penalty parameter λ) that governs the tradeoff between how well-conditioned the fitting problem will be and the increase in bias that naturally comes from looking at a smaller class of models. Choosing this tradeoff intelligently may be rather more important than the specific choice of regularization strategy. A detailed discussion of how to make this tradeoff is beyond the scope of the class; but we will see some of the computational tricks involved in implementing specific strategies for choosing regularization parameters before we are done.

7 Iterative methods

We started with a discussion of *direct* methods for solving least squares problems based on matrix factorizations. These methods have a well-understood running time, and they produce a solution that is accurate except for round-off effects. For larger or more complicated problems, though, we turn to *iterative* methods that produce a series of approximation solutions.

We will turn now to iterative methods: gradient and stochastic gradient approaches, Newton and Gauss-Newton, and (block) coordinate descent. We will see additional solver ideas as we move through the class, but these are nicely prototypical examples that illustrate two running themes in the design of numerical methods for optimization.

Fixed point iterations All our nonlinear solvers (and some of our linear solvers) will be *iterative*. We can write most as *fixed point iterations*

$$x^{k+1} = G(x^k), \tag{1}$$

which we hope will converge to a fixed point, i.e. $x^* = G(x^*)$. We often approach convergence analysis through the *error iteration* relating the error $e^k = x^k - x^*$ at successive steps:

$$e^{k+1} = G(x^* + e^k) - G(x^*). \tag{2}$$

Model-based methods Most nonquadratic problems are too hard to solve directly. On the other hand, we can *model* hard nonquadratic problems by simpler (possibly linear) problems as a way of building iterative solvers. The most common tactic — but not the only one! — is to approximate the nonlinear function by a linear or quadratic function and apply all the things we know about linear algebra. We will return to this idea in when we discuss Newton-type methods for optimization.

8 Gradient descent

One very simple iteration is *steepest descent* or *gradient descent*:

$$x^{k+1} = x^k - \alpha_k \nabla \phi(x^k) \tag{3}$$

where α_k is the *step size*, chosen adaptively or with some fixed schedule.

To understand the convergence of this method, consider gradient descent with a fixed step size α for the quadratic model problem

$$\phi(x) = \frac{1}{2}x^T Ax + b^T x + c$$

where A is symmetric positive definite. We have computed the gradient for a quadratic before:

$$\nabla\phi(x) = Ax + b,$$

which gives us the iteration equation

$$x_{k+1} = x_k - \alpha(Ax_k + b).$$

Subtracting the fixed point equation

$$x_* = x_* - \alpha(Ax_* + b)$$

yields the error iteration

$$e_{k+1} = (I - \alpha A)e_k.$$

If $\{\lambda_j\}$ are the eigenvalues of A, then the eigenvalues of $I - \alpha A$ are $\{1 - \alpha\lambda_j\}$. The spectral radius of the iteration matrix is thus

$$\max\{|1 - \alpha\lambda_j|\}_j = \max\left(|1 - \alpha\lambda_{\min}|, |1 - \alpha\lambda_{\max}|\right).$$

The iteration converges provided $\alpha < 2/\lambda_{\max}$, and the optimal α is

$$\alpha_* = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

which leads to the spectral radius

$$1 - \frac{2\lambda_{\min}}{\lambda_{\min} + \lambda_{\max}} = 1 - \frac{2}{1 + \kappa(A)}$$

where $\kappa(A) = \lambda_{\max}/\lambda_{\min}$ is the condition number for the (symmetric positive definite) matrix A. If A is ill-conditioned, then, we are forced to take very small steps to guarantee convergence, and convergence may be heart breakingly slow. We will get to the minimum in the long run — but, then again, in the long run we all die.

9 The Benefits of Slow Convergence

How steepest descent behaves on a quadratic model is how it behaves generally: if x_* is a strong local minimizer of some general nonlinear ϕ, then gradient descent with a small enough step size will converge locally to x_*. But if $H_\phi(x_*)$ is ill-conditioned, then one has to take small steps, and convergence can be quite slow.

Somewhat surprisingly, sometimes we *want* this slow convergence. To illustrate why, consider the *Landweber iteration*, which is steepest descent iteration applied to linear least squares problems:

$$x^{k+1} = x^k - \alpha_k A^T (Ax^k - b).$$

If we start from the initial guess $x^0 = 0$ and let the step size be a fixed value $\alpha_k = \alpha$, we have the subsequent steps

$$\begin{aligned}
x^1 &= \alpha A^T b \\
x^2 &= (I - \alpha A^T A)\alpha A^T b + \alpha A^T b \\
x^3 &= (I - \alpha A^T A)^2 \alpha A^T b + (I - \alpha A^T A)\alpha A^T b + \alpha A^T b
\end{aligned}$$

and so forth. That is, each step is a partial sum of a *Neumann series*, which is the matrix generalization of the geometric series

$$\sum_{j=0}^{k} z^j = (1 - z^{k+1})(1 - z)^{-1} \to (1 - z)^{-1} \text{ as } k \to \infty \text{ for } |z| < 1.$$

Using the more concise expression for the partial sums of the Neumann series expansion, we have

$$\begin{aligned}
x^{k+1} &= \sum_{j=0}^{k} (I - \alpha A^T A)^j \alpha A^T b \\
&= (I - (I - \alpha A^T A)^{k+1})(\alpha A^T A)^{-1} \alpha A^T b \\
&= (I - (I - \alpha A^T A)^{k+1}) A^\dagger b.
\end{aligned}$$

Alternately, we can write the iterates in terms of the singular value decomposition with a filter for regularization:

$$x^{k+1} = V \tilde{\Sigma}^{-1} U^T b, \quad \tilde{\sigma}_j^{-1} = (1 - (1 - \alpha \sigma_j^2)^{k+1}))\sigma_j^{-1}.$$

Hence, rather than running the Landweber iteration to convergence, we typically stop when k is large enough so that the filter is nearly the identity for large singular values, but is small enough so that the influence of the small singular values is suppressed.

10 Gradient descent with errors

Before we turn to stochastic gradient descent, let us instead look at how to analyze *gradient descent with errors*. In particular, consider the iteration

$$x^{k+1} = x^k - \alpha_k p^k$$

where

$$p^k = \nabla \phi(x^k) + u^k$$

for some error u^k that is "small." As before, let's keep things simple by looking how this iteration behaves for a quadratic model problem with a fixed step size, i.e.

$$x^{k+1} = x^k - \alpha(Ax^k + b + u^k).$$

Subtracting x^* from both sides gives the error iteration

$$e^{k+1} = (I - \alpha A)e^k - \alpha u^k.$$

A little mumbling over the iteration gives us

$$e^{k+1} = (I - \alpha A)^{k+1} e^0 - \alpha \sum_{j=0}^{k} (I - \alpha A)^{k-j} u^j.$$

In order to analyze the second term in this iteration, we need some additional sort of control. In the simplest case, that control might be deterministic. For example, if we can guarantee that $\|u^k\| \leq C\gamma^{-k}$, then we have the bound

$$\left\| \sum_{j=0}^{k} (I - \alpha A)^{k-j} u_j \right\| \leq C\gamma^{-k} \sum_{j=0}^{k} (\gamma \|(I - \alpha A)\|)^{k-j} \leq \frac{C\gamma^{-k-1}}{1 - \gamma \|I - \alpha A\|}.$$

Hence, we can make the iteration converge with inaccurate gradients, as long as the accuracy improves sufficiently quickly with time.

We will pick up this iteration again next time under the assumption that the errors are random, which is what happens in the *stochastic gradient* method.