2019-06-05

1 Regularization in kernel methods

Last time, we discussed kernel methods for *interpolation*: given f_X, we seek an approximation $\hat{f}_X$ such that $\hat{f}_X = f_X$. However, the kernel matrix K_{XX} is sometimes very nearly ill-conditioned, particularly when the underlying kernel is smooth (e.g. a squared exponential kernel). This near-singularity means that when f is not smooth enough to lie in the native space $\mathcal{H}$ for the kernel, or when our measurements are contaminated with some error, kernel interpolation schemes may not give accurate results. And even when f is in the appropriate native space and we are able to evaluate f exactly, this lack of stability may cause problems because of the influence of rounding errors.

We deal with the problem of instability in kernel methods the same way we deal with instability in other fitting problems: we *regularize*. There are many ways that we might choose to regularize, but we will focus on the common Tikhonov regularization approach. As always in kernel methods, there are multiple stories for the same method; we will tell two of them.

1.1 Feature space and kernel ridge regression

Recall the feature space version of kernel interpolation: write

$$\hat{f}(x) = \psi(x)^T c$$

where c is determined by the problem

$$\text{minimize } \|c\|^2 \text{ s.t. } \Psi^T c = f_X$$

with Ψ the matrix whose columns are feature vectors at the data points in X. *Kernel ridge regression* instead solves the unconstrained minimization problem

$$\text{minimize } \lambda\|c_\lambda\|^2 + \|\Psi^T c_\lambda - f_X\|^2$$

That is, rather than enforcing the interpolation constraints, we minimize a combination of a regularity term (the norm of c_λ) and a data fidelity term. As the weight λ goes to zero, we recover kernel interpolation. For nonzero λ, we solve the critical point equations

$$\lambda c_\lambda + \Psi(\Psi^T c_\lambda - f_x) = 0,$$

which we may rewrite using $r = \Psi^T c_\lambda - f_X$ as

$$\begin{bmatrix} \lambda I & \Psi \\ \Psi^T & -I \end{bmatrix} \begin{bmatrix} c_\lambda \\ r \end{bmatrix} = \begin{bmatrix} 0 \\ f_X \end{bmatrix}.$$

Eliminating c_λ gives the equation

$$-(I + \lambda^{-1} \Psi^T \Psi) r = f_X,$$

and back-subsittution yields

$$\lambda c_\lambda = \Psi (I + \lambda^{-1} \Psi^T \Psi)^{-1} f_X.$$

Dividing both sides by λ, we have

$$\hat{f}_\lambda(x) = \psi(x)^T c_\lambda = \psi(x)^T \Psi (\Psi^T \Psi + \lambda I)^{-1} f_X,$$

and applying the kernel trick gives

$$\hat{f}_\lambda(x) = k_{xX} (K_{XX} + \lambda I)^{-1} f_X.$$

As with kernel interpolation, the story of kernel ridge regression can be told without reference to a particular basis or feature map. Observe as before that $\Psi^T c_\lambda = \hat{f}_{\lambda,X}$ and that $\|c_\lambda\| = \|\hat{f}\|_{\mathcal{H}}^2$ for an appropriate reproducing kernel Hilbert space. The kernel ridge regression problem is therefore

$$\text{minimize } \lambda \|\hat{f}\|_{\mathcal{H}}^2 + \|\hat{f}_X - f_X\|^2 \text{ over } \hat{f} \in \mathcal{H}.$$

1.2 GPs with noise

The interpretation of Tikhonov regularization for Gaussian processes is straightforward. Suppose that f is drawn from a GP with mean zero and covariance kernel K, and we wish to compute a marginal distribution conditioned on knowing $y = f_X + u$ where u is a vector of independent Gaussian random variables with zero mean and variance σ^2. Then the posterior distribution for f conditioned on the data is a GP with mean and covariance

$$\hat{\mu}(x) = k_{xX} \tilde{K}^{-1} f_X, \qquad \hat{k}(x, x') = k(x, x') - k_{xX} \tilde{K}^{-1} k_{Xx'}$$

where $\tilde{K} = K + \sigma^2 I$. The derivation comes from looking at the multivariate Gaussian distribution of the data (y) together with function values at locations of interest.

2 Choosing hyperparameters

Whether we call it kernel ridge regression or GP regression with noise, Tikhonov regularization involves a free parameter — which may come in addition to other hyper-parameters in the kernel, like length scale or scale factor. How do we choose all these hyper-parameters? There are several methods, though all sometimes fail, and none of them is uniformly the best. We will discuss four approaches: the discrepancy principle, the L-curve, cross-validation, and maximum likelihood estimation.

2.1 The discrepancy principle and the L-curve

The *discrepancy principle* (due to Morozov) says that we should choose the regularization parameter based on the variance of the noise. This may be useful when the primary source of error comes from measurement noise from instruments (for example), but often we do not have this information; what shall we do then? Also, the discrepancy principle does not tell us what to do with other hyper-parameters, such as kernel length scales.

The *L-curve* is a graphical plot on a log-log axis of the norm of the data fidelity term versus the norm of the regularization term. Often such plots have a "corner" associated with the favored choice λ_* for the regularization parameter. Increasing λ beyond λ_* increases the residual error quickly, while decreasing λ from λ_* has only a modest impact on the residual error, but causes a rapid increase in the size of the regularization term. As with the discrepancy principle, the L-curve is primarily used for determining a single regularization parameter, and not for fitting other hyper-parameters such as the kernel length-scale.

2.2 Cross-validation

The idea of *cross-validation* is to fit the method to split the training data into two sets: a subset that we actually use to fit the model, and a held-out set to test the generalization error. We usually use more than one splitting of the data to do this. For example, the *leave-one-out cross-validation* (LOOCV) statistic for a regression method for a function f on data points X is

$$\mathrm{LOOCV} = \frac{1}{n} \sum_{i=1}^{n} (f^{(-i)}(x_i) - f(x_i))^2$$

where $f^{(-i)}$ refers to the model fit to all of the points in X except for x_i. One can do more complicated things, but the LOOCV statistic has a lovely structure that lets us do fast computations, and this is worth exploring.

2.2.1 Fast LOOCV for least squares

Before we explore the case of kernel methods, let us first consider LOOCV for the ordinary linear least squares problem:

$$\text{minimize } \|Ax - b\|^2$$

To compute the LOOCV statistic in the most obvious way, we would delete each row a_i^T of A in turn, fit the model coefficients $x^{(-i)}$, and then evaluate $r^{(-i)} = b_i - a_i^T x^{(-i)}$. This involves m least squares problems, for a total cost of $O(m^2n^2)$ (as opposed to the usual $O(mn^2)$ cost for an ordinary least squares problem). Let us find a better way!

The key is to write the equations for $x^{(-i)}$ as a small change to the equations for $A^TAx^* = A^Tb$:

$$(A^TA - a_ia_i^T)x^{(-i)} = A^Tb - a_ib_i.$$

This subtracts the influence of row i from both sides of the normal equations. By introducing the auxiliary variable $\gamma = -a_i^T x^{(-i)}$, we have

$$\begin{bmatrix} A^TA & a_i \\ a_i^T & 1 \end{bmatrix} \begin{bmatrix} x^{(-i)} \\ \gamma \end{bmatrix} = \begin{bmatrix} A^Tb - a_ib_i \\ 0 \end{bmatrix}.$$

Eliminating $x^{(-i)}$ gives

$$(1 - \ell_i^2)\gamma = \ell_i^2 b_i - a_i^T x^*$$

where $\ell_i^2 = a_i^T(A^TA)^{-1}a_i$ is called the *leverage score* for row i. Now, observe that if $r = b - Ax^*$ is the residual for the full problem, then

$$(1 - \ell_i^2)r^{(-i)} = (1 - \ell_i^2)(b_i + \gamma) = (1 - \ell_i^2)b_i + \ell_i^2 b_i - a_i^T x_* = r_i,$$

or, equivalently

$$r^{(-i)} = \frac{r_i}{1 - \ell_i^2}.$$

We finish the job by observing that ℓ_i^2 is the ith diagonal element of the orthogonal projector $\Pi = A(A^TA)A^{-1}$, which we can also write in terms of the economy QR decomposition of A as $\Pi = QQ^T$. Hence, ℓ_i^2 is the squared row sum of Q in the QR factorization.

2.2.2 Fast LOOCV for kernels

The trick for computing the LOOCV statistic for kernels is similar to the trick for least squares, at least in broad outlines. Let c be the coefficient vector fit to all the nodes, and let $c^{(-i)}$ be the coefficient vector for the expansion fit to all the nodes except node i; that is, we want $c_i^{(-i)} = 0$ and we allow $r^{(-i)} = f(x_i) - k_{xX}c^{(-i)}$ to be nonzero. Then

$$\begin{bmatrix} \tilde{K} & e_i \\ e_i^T & 0 \end{bmatrix} \begin{bmatrix} c^{(-i)} \\ r^{(-i)} \end{bmatrix} = \begin{bmatrix} f_X \\ 0 \end{bmatrix},$$

and Gaussian elimination of $c^{(-i)}$ yields

$$[\tilde{K}^{-1}]_{ii} r^{(-i)} = e_i^T \tilde{K}^{-1} f_X = c_i,$$

and therefore

$$r^{(-i)} = \frac{c_i}{[\tilde{K}^{-1}]_{ii}}.$$

The observent reader may notice that this yields essentially the same argument we saw in the error analysis of kernel methods, and that $[\tilde{K}^{-1}]_{ii}^{-1}$ is the squared power function for evaluating the error at x_i given data at all the other points.

What about the derivatives of $r^{(-i)}$ with respect to any hyper-parameters? After all, these are important if we are going to optimize. We know that

$$\delta[\tilde{K}^{-1}] = -\tilde{K}^{-1}[\delta\tilde{K}]\tilde{K}^{-1}$$

and differentiating $c = K^{-1}f_X$ (and using the fact that the function values are independent of the hyper-parameters) gives

$$\delta c = -\tilde{K}^{-1}[\delta\tilde{K}]c.$$

Let w denote $\tilde{K}^{-1}e_i$; then

$$\delta c_i = -w^T[\delta\tilde{K}]c, \qquad \delta\left([K^{-1}]_{ii}\right) = -w^T[\delta\tilde{K}]w,$$

and the quotient rule, together with a little algebra, gives

$$\delta r^{(-i)} = \frac{[\delta\tilde{K}w]^T(wc_i - cw_i)}{[\tilde{K}^{-1}]_{ii}^2}.$$

2.3 Maximum likelihood estimation

Finally, we consider the *maximum likelihood estimation* scheme. If we have data y drawn from a distribution $p(y;\theta)$ where θ are unknown parameters, the idea of maximum likelihood is to maximize $p(y;\theta)$ with respect to θ. Often the probability density is awkwardly scaled for computation, and so we typically instead use the *log* likelihood

$$\mathcal{L}(\theta) = \log p(y;\theta)$$

In the case of Gaussian processes, we have

$$p(y) = \frac{1}{\sqrt{\det(2\pi K)}} \exp\left(-\frac{1}{2}(y-\mu)^T K^{-1}(y-\mu)\right)$$

and

$$\log p(y) = -\frac{1}{2}(y-\mu)^T K^{-1}(y-\mu) - \frac{1}{2}\log\det(K) - \frac{n}{2}\log(2\pi)$$

The first term is the *model fidelity* term; it is larger the closer y is to μ in the norm induced by K^{-1}, with a maximum value of zero when $y = \mu$. The second term is the *model complexity* term; it is larger when K has lower volume, i.e. the likely model predictions are in a smaller region. The last term is independent of any kernel hyper-praameters, and so is irrelevant for optimization.

With an eye to optimization, we again want to compute derivatives. The derivative of the model fidelity term with respect to kernel hyperparameters is straightforward:

$$\delta\left[-\frac{1}{2}(y-\mu)^T K^{-1}(y-\mu)\right] = \frac{1}{2}c^T[\delta K]c$$

where $c = K^{-1}(y-\mu)$. The more interesting piece is the derivative of the log determinant. To get this, we observe that

$$\det(I+E) = \prod_{i=1}^n (1+\lambda_i(E)),$$

and if E is small, linearization about $E = 0$ gives

$$\det(I+E) = 1 + \sum_{i=1}^n \lambda_i(E) + O(\|E\|^2).$$

Therefore, the derivative of the determinant at the identity in a direction E is just $\operatorname{tr}(E) = \sum_i \lambda_i(E) = \sum_i E_{ii}$. We deal with the more general case by observing that $\det(K+E) = \det(K)\det(I + K^{-1}E)$; hence,

$$\delta[\det(K)] = \det(K)\operatorname{tr}(K^{-1}\delta K).$$

Finally, we have

$$\delta[\log\det(K)] = \frac{\delta[\det(K)]}{\det(K)} = \operatorname{tr}(K^{-1}\delta K).$$