Loss-Penalty Estimators

Ethan Ligon

April 26, 2023

High Dimensional Prediction/"Modern Inference"

Back to the problem of estimating

$$m(x) = \mathbb{E}(y|x),$$

the conditional moment function. Note that here y is random, but that x is fixed (though it may be a realization of an rv).

- In this setting this problem is all about *prediction*; we're indifferent to whether x is exogenous or not.

Approaches to non-parametric estimation

x low dimensional Use kernel regression

x high dimensional "Modern inference"; a variety of machine learning tools (our topic for today). This can be:

- Parametric; or
- Non-parametric

Loss-Penalty Form

A really wide variety of estimators can be written in so called "loss-penalty" form, where we try to choose a vector of parameters b to solve

$$\min_{b \in B} L(b) + \lambda P(b).$$

The first term $L(b)$ is something like (minus) a log-likelihood, or the GMM criterion, or some other loss function. The second term $P(b)$ is a "penalty" function. Typically this will introduce some cost associated with more/larger parameters. The parameter $\lambda > 0$ is a "tuning" parameter, which allows one to balance bias vs variance; larger values "penalize" large b more, increasing bias so as to reduce variance.

Various Penalizations

A variety of familiar approaches to trying to encourage models with fewer parameters: goal is to achieve balance between bias and variance.

Examples

- Adjusted R^2: $1 - (1 - R^2)\frac{N-1}{N-k-1}$
- Akaike Information Criterion: $N(1 + \log 2\pi\hat{\sigma}^2) + 2k$
- Bayesian Information Criterion: $N(1 + \log 2\pi\hat{\sigma}^2) + k \log N$

Effective Degrees of Freedom

Consider a regression linear in p parameters; then the model can be represented as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{a} + \boldsymbol{e}.$$

Let $\boldsymbol{V} = \frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X} - \bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^\top$ be the sample covariance matrix of $\boldsymbol{X}$, and let d_j^2 be the eigenvalues of this matrix. Then the "effective degrees of freedom" for the regression in Loss-Penalty form is

$$df(\lambda) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}.$$

Least Squares Example

Think of our most basic linear least squares problem. We have $\boldsymbol{X}$, an $N \times p$ matrix, and $\boldsymbol{y}$, an $N \times 1$ vector. Then we have the loss function

$$L(b|\boldsymbol{X}, \boldsymbol{y}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}b)^\top(\boldsymbol{y} - \boldsymbol{X}b);$$

this gives rise to the familiar normal (first order conditions)

$$\boldsymbol{X}^\top \boldsymbol{X}\hat{b} = \boldsymbol{X}^\top \boldsymbol{y}.$$

We can solve for $\hat{b}$ if $\boldsymbol{X}$ has full column rank—a necessary condition for this is $N \geq p$.

Penalizing Least Squares

Now consider adding a penalty term to the objective function, say $P(b) = \lambda b^\top b = \lambda \|b\|^2$. Then the problem becomes

$$\min_b L(b|\boldsymbol{X}, \boldsymbol{y}) + P(b) = \min_b \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}b)^\top(\boldsymbol{y} - \boldsymbol{X}b) + \lambda\|b\|^2.$$

Now the FOC become

$$(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})\hat{b} = \boldsymbol{X}^\top\boldsymbol{y}.$$

Now we can solve for $\hat{b}$ even if $p > N$! (This is the so-called "ridge" estimator.)

The Lasso (Least Absolute Shrinkage and Selection Operator)

The Lasso takes the form

$$\min_{b \in B} \sum_{j=1}^{N} (y_j - X_j b)^2 + \lambda \sum_{k=1}^{K} |b_k|$$

The absolute value penalty (L_1 norm) means that the method will try to set coefficients to zero where doing so doesn't compromise the fit too much. Thus, the larger λ the fewer non-zero coefficients we expect to see (think, the more parsimonious the regression specification).

Tuning

So, how should we choose λ? Too small, and we increase bias; too big we increase variance. Note that in the Lasso case choosing *one* parameter can have the effect of introducing or eliminating *lots* of parameters.

Cross-Validation

The cross-validation tools we discussed last time have many uses, but one very simple and effective use case involves tuning just a single parameter to try and minimize MSE. Let

$$\text{CV}(\lambda) = \frac{1}{N}\sum_{j=1}^{N} e_{-j}(\lambda)^2;$$

Then choose

$$\lambda^* = \underset{\lambda \in \mathbb{R}_+}{\arg\min}\, \text{CV}(\lambda).$$

Linear-Nonlinear, Again

Last lecture we introduced the idea of representing functions by using a kernel to construct a basis for a vector space. We gave an example in which we used a "radial basis function" to construct such a basis, with

$$K(x, c_k) = e^{-\frac{1}{2}\|x - c_k\|^2},$$

where x, c_k are both in $\mathbb{R}^p$. We computed an example in which this basis seemed to work pretty well for representing a function

$$f(x) = \sum_k \alpha_k K(x, c_k),$$

and we used cross-validation to choose the number of terms in this sum.

Gram Matrix

Given data $x_1, \ldots, x_N$, recollect our definition of the Gram Matrix:

$$\boldsymbol{K} = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_N) \\ K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_N, x_1) & K(x_N, x_2) & \cdots & K(x_N, x_N) \end{bmatrix}$$

Notice that with the Gaussian RBF choice of K the Gram matrix also has a "ridge" along the diagonal.

Mercer Kernels

The RBF is an example of what are called *Mercer* kernels. Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^p$.
A Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:

- Is continuous;
- Is symmetric $(K(x, y) = K(y, x))$;
- Is positive semidefinite:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} K(x_i, x_j) c_i c_j \geq 0$$

for any finite sequence $\{x_i \in \mathcal{X}\}_{i=1}^{n}$ and any sequence of real numbers $\{c_i\}_{i=1}^{n}$.

Basis for a Function Space

Choose a kernel K, and let $K_{x_j}(x) = K(x_j, x)$ (we've used this trick before). Then we can use $K_{x_1}, K_{x_2}, \ldots K_{x_k}$ as the basis for functions, i.e.,

$$f(x) = \sum_{j=1}^{k} \alpha_k K_{x_j}(x) \qquad \text{(Note that } k \text{ can be infinite)},$$

or

$$g(y) = \sum_{j=1}^{\ell} \beta_\ell K_{y_j}(x).$$

Let

$$\mathcal{H} = \left\{ f : \sum_{j=1}^{k} \alpha_k K_{x_j}(x) \qquad \text{for some } k \right\}$$

be the set of all such functions.

NB

Note that K_{x_j} is itself trivially in $\mathcal{H}$.

Kernels, Norms, Inner Products

Inner Product

For any two functions f, g in $\mathcal{H}$, with $f(x) = \sum_{j=1}^{k} \alpha_k K_{x_j}(x)$ and $g(x) = \sum_{j=1}^{\ell} \beta_\ell K_{y_j}(x)$, define an inner product by

$$\langle f, g \rangle_K = \sum_{i=1}^{k} \sum j = 1^{\ell} \alpha_i \beta_j K(x_i, y_j).$$

Norm

The inner product implies a norm:

$$\|f\|_K = \sqrt{\langle f, f \rangle_K} = \sqrt{\sum_{i=1}^{k} \sum j = 1^{k} \alpha_i \alpha_j K(x_i, x_j)} = \sqrt{\alpha^\top \boldsymbol{K} \alpha},$$

where α is the column vector $(\alpha_1, \ldots, \alpha_N)^\top$ and $\boldsymbol{K}$ is the $k \times k$ matrix with the i, jth element given by $K(x_i, x_j)$.

The Reproducing Property

Let $f(x) = \sum_{j=1}^{k} \alpha_k K_{x_j}(x) \in \mathcal{H}$. Using the inner product we just defined, we have

$$\langle f, K_x \rangle_K = \sum_{j=1}^{k} \alpha_k K(x_j, x) = f(x),$$

and that

$$\langle K_x, K_x \rangle_K = K(x, x).$$

This is called the *reproducing kernel property*. Any Mercer kernel has this property.

Reproducing Kernel Hilbert Space (RKHS)

An RKHS can be thought of as a class of smooth functions defined by a Mercer kernel. Choose such a kernel K. Endow the function space $\mathcal{H}$ with the inner product $\langle \cdot, \cdot \rangle_K$ (now it's a Hilbert space) and the norm $\| \cdot \|_K$ (now it's also normed vector space).

Why should we care about RKHS?

If we choose a Mercer kernel K, we can use this to define an RKHS. But the functions in this space still look pretty special! How does this help us represent some smooth function which isn't in this set?

Hand-wavy Answer

$\mathcal{H}_K$ is dense in the space of bounded continuous functions, and convergence is uniform.

RKHS Equivalence

For any kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is continuous & symmetric the following are equivalent:

1. Every Gram matrix is positive semi-definite.
2. The kernel K is the reproducing kernel of an RKHS of functions on $\mathcal{X}$.

Loss-Penalty Criteria

We want to estimate an unknown function f^* using data $\boldsymbol{D}_N = \{(y_1, x_1), (y_2, x_2), \ldots, (y_N, x_N)\}$.

- As usual, we can think about a loss function that depends on the data and also on our candidate function (say f), so that we might estimate, say,

$$\hat{f} \in \arg\min_f \mathcal{L}(f; \boldsymbol{D}_N).$$

But this won't do—if we can just choose whatever f we want we'll just choose one that interpolates all the data points!

Penalty (Regularizer)

One solution to this is to introduce a "penalty" function, or "regularizer", which imposes a penalty for more "complicated" functions.

An estimation problem uses a "loss-penalty" criterion if it can be expressed in the form

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \mathcal{L}(\boldsymbol{D}_N, \boldsymbol{F}_N) + \mathcal{R}(\|f\|_K^2).$$

Representer Theorem

We want to estimate a continuous, bounded function f^*.

1. Choose a Mercer kernel K and let $\mathcal{H}_K$ be the corresponding "Reproducing Kernel Hilbert Space".
2. Let $\mathcal{L}(f|D_N)$ be a loss function which maps data $\boldsymbol{D}_N = \{(y_1, x_1), (y_2, x_2), \ldots, (y_N, x_N)\}$ and candidate f into $\mathbb{R}$.
3. Let $\mathcal{P} : \mathbb{R} \to \mathbb{R}$ be any increasing function.

Representer Theorem

If the estimation problem can be expressed in the loss-penalty form

$$\hat{f} = \arg\min_{f \in \mathcal{H}_K} \mathcal{L}(\boldsymbol{D}_N, \boldsymbol{F}_N) + \mathcal{R}(\|f\|_K^2)$$

with $\mathcal{R}$ strictly increasing, then any solution will have the form

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x).$$

RKHS Regression

Back to estimating the CEF, $m(x) = \mathbb{E}(y|x)$, but now with x possibly high dimensional. We want to solve

$$\hat{m} = \arg\min_{m \in \mathcal{H}_K} \sum_{i=1}^{N} (y_i - m(x_i))^2 + \lambda \|m\|_K^2.$$

By the representer theorem, $\hat{m}(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x)$. Substitute into the objective function, get ("kernel trick")

$$\hat{m} = \arg\min_{\alpha \in \mathbb{R}^N} (\boldsymbol{y} - \boldsymbol{K}\alpha)^\top (\boldsymbol{y} - \boldsymbol{K}\alpha) + \lambda \alpha^\top \boldsymbol{K} \alpha.$$

The solution to this is just like the ridge regression example we gave earlier, with normal equations

$$(\boldsymbol{K}^\top \boldsymbol{K} + \lambda \boldsymbol{K})\hat{\alpha} = \boldsymbol{K}^\top \boldsymbol{y}.$$

But here $\boldsymbol{K}$ is symmetric & invertible, so

$$\hat{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}.$$

Predictions, Errors

We're not done! First, we don't really care about α *per se*; our interest is in

$$\hat{m}(x) = \sum_j \hat{\alpha}_j K_{x_j}(x).$$

Second, our RKHS regression estimator still depends on the tuning parameter λ. As in our earlier case we can address this using cross-validation, for which we only care about $\hat{m}$ evaluated at the N data points

$$\hat{\boldsymbol{y}} = \boldsymbol{K}\hat{\alpha} = \boldsymbol{K}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} = \boldsymbol{P}_K(\lambda)\boldsymbol{y},$$

with $\boldsymbol{P}_K(\lambda)$ a projection matrix (for fixed λ). Then residuals are

$$\hat{\boldsymbol{e}} = \left(\boldsymbol{I} - \boldsymbol{K}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\right)\boldsymbol{y} = \boldsymbol{M}(\lambda)\boldsymbol{y},$$

with $\boldsymbol{M}(\lambda)$ a linear operator mapping $\boldsymbol{y}$ into residuals $\hat{\boldsymbol{e}}$.

Better Cross-Validation

Recall that our cross-validation criterion relied on averaging across N "leave-one-out" estimators. Here we're in the position of *not* needing to re-estimate everying N times, because for this linear mapping we can just delete the diagonal elements of the operator $\boldsymbol{M}(\lambda)$, obtaining

$$\hat{\boldsymbol{e}}_{-}(\lambda) = (\boldsymbol{M}(\lambda) - \operatorname{dg} \boldsymbol{M}(\lambda))\boldsymbol{y} = \boldsymbol{M}_{-}(\lambda)\boldsymbol{y}.$$

Thus the cross validation criterion is simply

$$\mathsf{CV}(\lambda) = \frac{1}{N}\hat{\boldsymbol{e}}_{-}(\lambda)^{\top}\hat{\boldsymbol{e}}_{-}(\lambda) = \frac{1}{N}\boldsymbol{y}^{\top}\boldsymbol{M}_{-}(\lambda)^{\top}\boldsymbol{M}_{-}(\lambda))\boldsymbol{y}.$$

Note that in contrast to what we did last lecture this cross-validation criterion is a *smooth* function of the continuous variable λ.