Chapter 12

Model Inference and Averaging

Most methods described up to know have been based on minimizing sums of square (regression) or minimizing cross-entropy (classification). Both of these are special cases of maximum likelihood estimation.

12.1 Maximum Likelihood Estimation

The bootstrap methods provide a direct computational way of assessing uncertainty, by sampling from the training data. Here we will illustrate the bootstrap in a simple one-dimensional smoothing problem and show its connection to maximum likelihood.

We are fitting a cubic spline to some data with some predetermined knots. The spline space is linear so we can parameterize it and write:

$$\mu(x) = \sum_{j=1}^{J} \theta_j b_j(x).$$

We can then write the lease squares estimate of β with the linear equation:

$$\hat{\beta} = (\mathbf{B}'\mathbf{B})^{-1}\mathbf{B}'\mathbf{y}$$

.

The estimated covariance structure is

$$\hat{\text{var}}(\hat{\beta}) = (\mathbf{B}'\mathbf{B})^{-1}\hat{\sigma}^2$$

where we estimate the noise variance σ^2 with

$$\hat{\sigma}^2 = \sum_{i=1}^{N} \{y_i - \hat{\mu}(x_i)\}^2/N.$$

We can create point-wise confidence intervals by taking $\hat{\mathbf{se}}[\hat{\mu}(x)]$ (obtained from the diagonal of the covariance matrix) and considering $\hat{\mu}(x) \pm 1.96 \hat{\mathbf{se}}[\hat{\mu}(x)]$. These are a confidence interval only if $\hat{\mu}(x)$ is normally distributed.

We can instead use the bootstrap as described in the previous section to obtain bootstrap confidence intervals. If we instead use the parametric bootstrap

$$\hat{y}_i^* = \hat{\mu}(x_i) + \epsilon_i, \epsilon_i \sim N(\hat{\mu}, \hat{\sigma}^2)$$

for practical purposes we get the distribution obtained above for the least squares regression

$$\hat{\mu}^*(x) \sim N(\hat{\mu}, \hat{\mathbf{se}}[\hat{\mu}(x)])$$

12.2 Maximum Likelihood Inference

It turns out parametric bootstrap agrees with least squares in the previous example because the model had Gaussian errors. In general the parametric bootstrap agrees not with least squares but with maximum likelihood.

For maximum likelihood estimation we start with a probability density function for our observations

$$z_i \sim g_\theta(z).$$

In this expression θ represents one or more unknown parameters that govern the distribution of Z. For example we assume a Gaussian distribution then

$$\theta = (\mu, \sigma^2),$$

and

$$g_\theta(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{1}{2}(z-\mu)^2/\sigma^2}$$

Maximum likelihood is based on the likelihood function, given by

$$L(\theta; \mathbf{Z}) = \prod_{i=1}^{N} g_\theta(z_i)$$

the probability of the observed data under the model g_θ.

We usually work with the logarithm or the log likelihood

$$\begin{aligned} l(\theta; \mathbf{Z}) &= \sum_{i=1}^{N} l(\theta; z_i) \\ &= \sum_{i=1}^{N} \log g_\theta(z_i) \end{aligned}$$

The maximum likelihood estimate chooses the θ that minimizes $l(\theta; \mathbf{Z})$. Notice that this value has the property that the derivative

$$\sum_{i=1}^{N} \dot{l}(\theta; z_i) = 0$$

There is theory that gives us asymptotic distribution of this estimate. The above sum, Taylors expansion and the CLT are used to show it is asymptotically normal.

12.2.1 Bootstrap and MLE

The bootstrap is sort of a computer implementation of nonparametric or parametric maximum likelihood. An advantage of the bootstrap is that it permits us to compute maximum likelihood estimates of standard errors and other quantities when no closed form solutions are available.

For example, consider the B-spline problem discussed above. If we chose the knots with some automatic procedure and wanted to include the variation introduced by this data-driven procedure, it would be very difficult to obtain closed form solutions for the standard error s of our estimates. Using the bootstrap we can get these.

12.3 Bayesian Methods

In the Bayesian framework we specify a sampling distribution for our data $\Pr(Z|\theta)$ (as we have done all along) but we also specify a *prior* distribution $\Pr(\theta)$ for our parameter. We can think of $\Pr(\theta)$ as a probabilistic quantity reflecting our prior knowledge about the parameter of interest. Instead of a maximum likelihood estimate we compute a posterior probability of the parameter (which is now a random variable):

$$\Pr(\theta|\mathbf{Z}) = \frac{\Pr(\mathbf{Z}|\theta)\Pr(\theta)}{\int \Pr(\mathbf{Z}|\theta)\Pr(\theta)\,d\theta}.$$

This formula is derived using Bayes theorem and represents the updated information about the parameters given we have observed data.

Notice that now instead of p-values and confidence intervals we can actually talk about the probability of θ falling in a particular interval.

The posterior distribution also provides a way to predict a new observation z^{new}:

$$\Pr(z^{new}|\mathbf{Z}) = \int \Pr(z^{new}|\theta) \Pr(\theta|\mathbf{Z})\, d\theta$$

In contrast, the maximum likelihood approach would predict using the distribution $\Pr(z^{new}|\hat{\theta})$ which does not account for the uncertainty in estimating θ.

Under a certain Bayesian framework we obtain the same solution to the spline example as with MLE. The idea is that assume the coefficients that define the spline are random and correlated (the correlation imposes smoothness). See the book for more details.

Also, there is a connection between the bootstrap and Bayesian Inference.

12.3.1 MCMC algorithm

Sometimes there is no closed form solution to

$$\Pr(\theta|\mathbf{Z}) = \frac{\Pr(\mathbf{Z}|\theta) \Pr(\theta)}{\int \Pr(\mathbf{Z}|\theta) \Pr(\theta)\, d\theta}.$$

In this case it is hard or impossible to compute quantities of interest such as the mean E($\theta|\mathbf{Z}$), the mode, the 2.5% and % 97.5, etc...

However, if we were able to sample from this posterior distribution we could, for example, take the average of 1000 samples to get a good estimate of the mean. Similarly for any other quantity that is a function of the posterior distribution.

But sampling is not easy either. MCMC algorithms provide a way to do this.

Monte Carlo

A Monte Carlo simulation can be defined as the use of random (or more realistically pseudo-random numbers) to solve a computational problem. For example is we wanted to integrate $\int_a^b h(x), dx$ we could simulate N uniform points x_i in (a, b) and use $\frac{1}{N}\sum_{i=1}^N h(x_i)$. This would give us an unbiased estimate. And if N is very large it would be precise as well.

Markov Chain

Loosely defined a Markov Chain is a stochastic process (random across time) that is completely defined by the *transition probabilities* $\Pr(X_{t+1}|X_t)$. Notice that X is a discrete process with K possible outcomes, the transition probabilities are defined by $K \times K$ table where entry i, j tells us the probability of going from state i to state j.

Markov Chains converge to a stationary state so that in the limit the distribution of X_t does not depend on t.

MCMC

Monte Carlo Markov Chain simulations use samples from a Markov Chain to solve numerical problems.

In turns out we can construct Markov Chains such the marginal distribution is the posterior distribution we are interested in sampling from. There are also some theorems that tell us that

$$\frac{1}{N}\sum_{n=1}^{N} h(X_n)$$

converges to E$[h(X)]$... which is good news.

Gibbs Sampler

There are various algorithms that get us what we want. A popular one is the Gibbs sampler. The algorithm is as follows:

1. Take some initials values $X_k^{(0)}$, k=1,...,K
2. Repeat for $t = 1, 2, \dots$
   Fork=1,2...,K generate $X_k^{(t)}$ from
   $$\Pr(X_k^{(t)} | X_{k-1}^{(t)}, \dots, X_{k-1}^{(t)}, X_{k+1}^{(t-1)}, \dots, X_K^{(t-1)})$$
3. Continue step 2 until the joint distribution of $(X_1^{(t)}, \dots, X_K^{(t)})$ doesn't change.

Then, for example, to get the mean we can simply take the sample mean.

Note: For the first few iterations there is dependence on the initial values which contradicts the stationarity that is needed. In practice we ignore the first few iterations (we call this *burn-in*).

12.4 EM

No time... see the book

12.5 Bagging and Bumping

To do bagging we perform bootstrap and for each bootstrap sample we keep the predictor $\hat{f}^{*b}(x)$. Then we form a new predictor based on the average of the bootstrap predictor

$$\hat{f}_{bag} = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^{*b}(x).$$

For linear smoothers and predictors this usually averages out to the original predictor, but for trees (next Chapter) it can help immensely.

In Bumping we pick the bootstrap predictor that minimizes the prediction error on the training sample. This sometimes help pick a version of the predictor not affected by outliers.