Statistics 360: Advanced R for Data Science

MARS, part III

Brad McNeney

Topics

- Recap of forward stepwise MARS algorithm (Algorithm 2)
- Pruning basis functions in the backward stepwise algorithm (Algorithm 3)
  - LOF revisited
- Software features
- Data structures and functions

Recap of forward algorithm

- The MARS forward stepwise algorithm (Algorithm 2) builds a linear prediction equation that is linear in basis functions $B_1(x), \ldots, B_{M_{max}}$.
  - Coefficients of the basis functions are by least squares.
  - Include an intercept term $B_0(x) = 1$ (notation change from Alg 2)
  - Basis functions are products of hinge functions:

$$B_m(x) = \prod_{k=1}^{K_m} h(s_{km}(x_{v(k,m)} - t_{km})),$$

  where
  - K_m is the number of product terms,
  - $h(x) = \max(0, x)$,
  - s_{km} is $+1$ or -1 (recall mirror-image basis functions),
  - $v(k, m)$ is the kth variable used in B_m, and
  - t_{km} is the knot for the kth variable.

Over-fitting

- During the forward algorithm, we added the basis function that improved LOF, the residual sum of squares (RSS)

$$\sum_{i=1}^{N}(y_i - \hat{f}_M(x_i))^2$$

  where $\hat{f}_M$ is a fitted model with M basis functions.
- RSS is OK for selecting among models with the same M, but not for comparing models with different M.
  - RSS decreases as we add predictors, even those not truly associated with the response, and so favours larger models.
- For model selection we need an unbiased measure of the "test error", which is the average squared error between observed and predicted values for data not used to fit the model.

Test error

- We call the RSS from an independent set of data not used to fit the model the validation error.
  - Split our data into "training" and "test" sets.
  - Estimate of test set error depends on split.
- Better: Use cross-validation (CV), which splits the data into "folds", fits the model on all but a hold-out, and averages the validation errors across folds.
- However, CV is time-consuming and so approximations are of interest.
- Generalized cross-validation, or GCV is one such approximation.

Generalized cross-validation (GCV)

- The LOF measure $LOF(\hat{f}(M)) = GCV(M)$ in Friedman's equations (30) and (32) is

$$\frac{1}{N}\frac{\sum_{i=1}^{N}(y_i - \hat{f}_M(x_i))^2}{(1 - \tilde{C}(M)/N)^2} = RSS \times \frac{N}{(N - \tilde{C}(M))^2}$$

  where $\tilde{C}(M) = C(M) + dM$, $C(M)$ is the sum of the hat-values from the fitted model and d is a smoothing parameter.
  - $C(M) = M + 1$ if there are no linear dependencies between basis functions, but summing the hat-values is safest.
  - Friedman suggests that $d = 3$ works well.
  - Denominator decreases, so GCV increases as M increases.
- Notice that for fixed M, and assuming no linear dependencies between basis functions, the best model is the one with smallest RSS, so our forward stepwise algorithm is OK as-is.
- Use GCV to compare models with different M in the backward stepwise algorithm.

Alternatives to GCV

- Backward stepwise selection is implemented in the R function `step()`. However, `step()` uses Mallow's C_p instead of GCV, where

$$C_p = RSS/S^2 - N + 2(M+1)$$

  for an estimate S^2 of σ^2 from a low-bias model, usually the largest one fit.
  - C_p is very similiar to Akaike's Information Criterion (AIC)
- As with GCV we see that C_p penalizes RSS with a penalty that becomes larger as M increases.
  - The factor of 2 in $2(M+1)$ can be modified to apply more/less penalty.
  - Replacing 2 with $\log(N)$ gives a Bayesian Information Criterion (BIC)-like penalty.

Backwards stepwise algorithm

- Initialize J^* to the set of all basis functions from the forward algorithm
- Outer loop over M in M_{max} to 2 (like `step()`):
  - Inner loop over M terms: Find the one that reduces GCV(M) the most (like `drop()`).
  - If GCV(M) best seen in outer loop, update J^*
- Algorithm terminates with best model J^*.