Forward selection for model building

Bernard Butler

2024-11-15

Working through an example

Perhaps the easiest way to see how forward selection works is by working through an example.

Let's say we have 4 candidate features $X = X[A, B, C, D]$, plus a constant term c, and we wish to predict a target y.

We also have three functions

1. $\texttt{fit}(X_{\text{train}}, y_{\text{train}})$,
2. $y_{\text{pred}} = \texttt{predict}(X_{\text{test}})$ and
3. $L = \texttt{lossMetric}(y_{\text{validation}}, y_{\text{pred}})$.

In practice, the advice is to compute L (the lossMetric) as the prediction error arising from a cross-validation procedure, rather than from a single validation set, or even the test set[1]. It should *never* be computed from the trainign set directly.

The procedure is as follows.

0. Starting with the constant model based on the feature $X_{\text{train}}[c]$, `predict` the associated y_{pred} and hence use `lossMetric` to compute $L(\texttt{constant})$.
1. Now add features A, B, C, D in turn to the constant term, fitting the training set and predicting the error using cross-validation, yielding $L(c, A)$, $L(c, B)$, $L(c, C)$, $L(c, D)$.
2. Let's say $L(c, A)$ gives the minimum error among the 4 models and is less than $L(c)$. Then feature A belongs in the model.
3. Now compute $L(c, A, B)$, $L(c, A, C)$, $L(c, A, D)$ using the cross-validation error estimate..
4. Let's say $L(c, A, C)$ gives the minimum error among the 3 models and is less than $L(c, A)$. Then features A, C belong in the model.
5. Now compute $L(c, A, C, B)$ and $L(c, A, C, D)$.

[1] A previous version of this guide suggested that the prediction error on the test set could be used as the loss metric. While this is a far better criterion than the prediction error on the training set, *its use is now discouraged*, because the cross-validation prediction error has been shown to be a better guide when choosing between feature subsets.

6. Let's say $L(c, A, C, D)$ gives the minimum error among the 2 models but is not less than $L(c, A, C)$. Then D does not belong in the model but we already know that features A, C belong in the model.
7. In that case, the best model we have found has features $\{c, A, C\}$ and we can ignore features B and D.

Note that the prediction should be chosen carefully. For regression, the mean square error is a popular choice; for classification, many metrics (accuracy, recall, F1-score, AUC, etc.) are available. The loss should always be computed with respect to data that is held back from training, such as the validation sets arising from cross-validation.

Explanation

As we added features one by one, we reduced the bias and increased the variance, but the total error continued to decrease until we had features A, C in the model. Beyond this point, the model overfitted the data: adding either B or D causes the total error to increase. This is because the variance increased more than the bias decreased. So our search stopped.

We did not save a lot of effort in this example, but more often than not, when the number of candidate features is large, forward selection stops after just a few have been chosen, because adding any of the remaining features would cause the loss metric to increase.

For comparison, a full enumeration over n features requires 2^n models to be built and checked. So, in the example above, we needed to fit 9 models instead of 16, and the advantage of forward selection ove brute foirce enumeration generally becomes even more apparent when the number of factors, n, grows. However, as with all *greedy* searches, it is not guaranteed to find the best model.

What this means to you

You need to write python code to do this, so that it can be run for each of the three targets. You should find that similar feature sets are chosen, but they are generally not identical.