Approximating Err and Cross-Validation

Stephen Vardeman

Analytics Iowa LLC

ISU Statistics and IMSE

Predictor complexity and Err

- Standard methods of predictor-making have associated "complexity/flexibility" parameters (like "*k*" for *k*-nn prediction or dimensionality of the input in MLR) to be chosen by an analyst

- One would like to choose these to minimize (the unknown) Err
  - too little complexity produces underfit and large prediction **bias**
  - Too much complexity produces overfit and large prediction **variance**

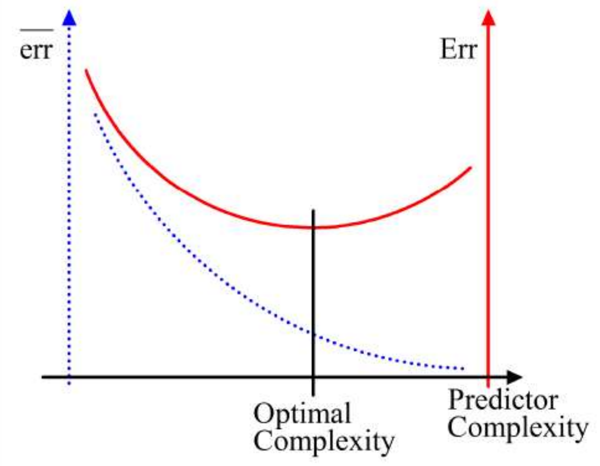
- All that is available for guiding an attempt to minimize Err is the training set, and measures that can be made from it

Training error

- The most obvious/elementary means of approximating Err is with the so-called "training error"

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L\left(\hat{f}\left(x_i\right), y_i\right)$$

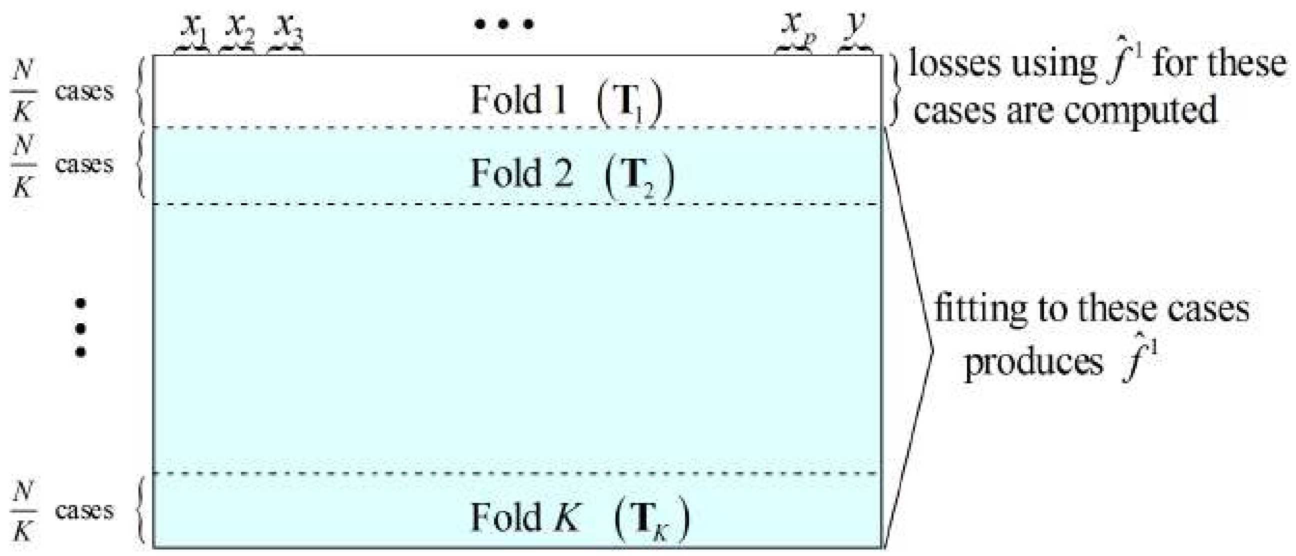
- But the training error is no good approximator of prediction error … typically one faces this reality:



- One cannot "train" and reliably "test" on the same data

Cross-validation

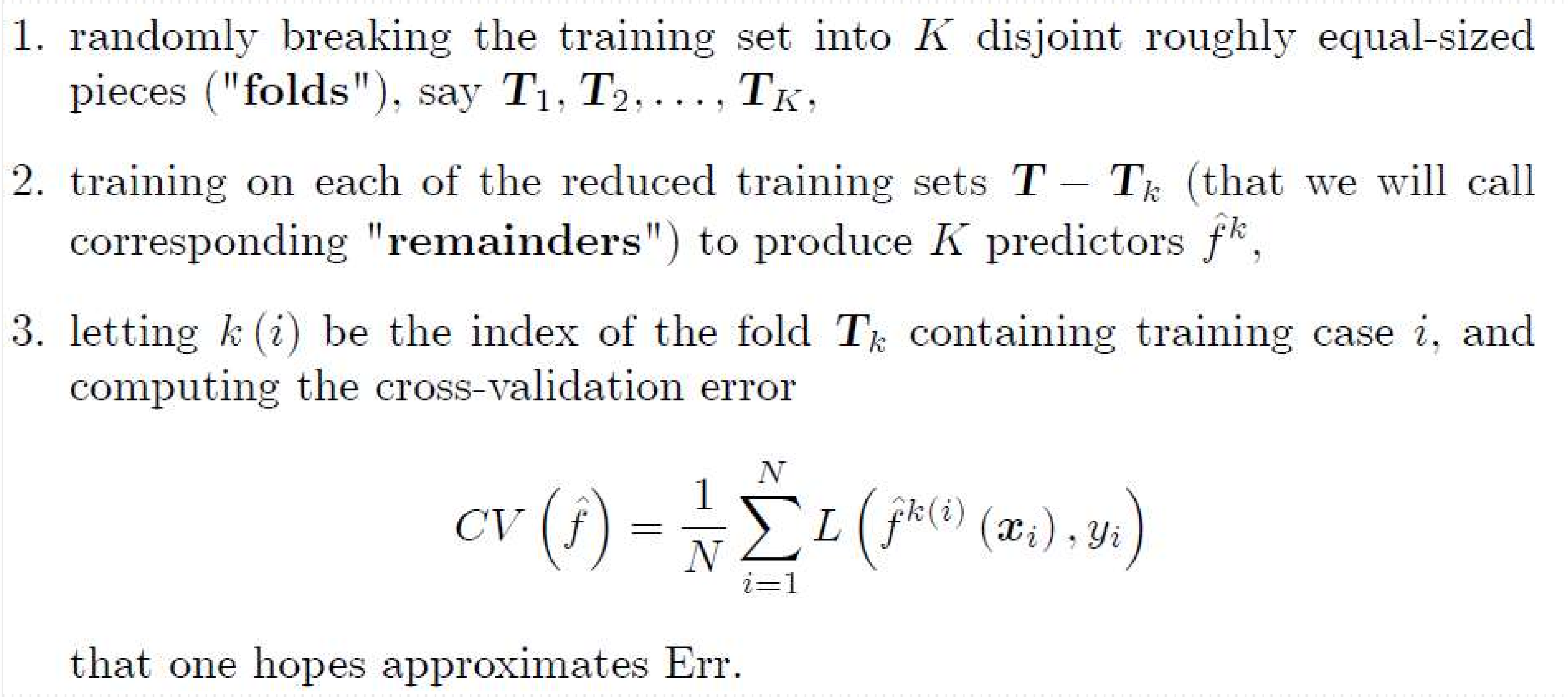
- The best existing method of inferring the likely effectiveness of a prediction methodology is through so-called “cross-validation”
- (For randomly ordered cases) this cartoon represents one (of *K*) similar steps in “*K*-fold” cross-validation:



x_1 x_2 x_3 • • • x_p y
$\frac{N}{K}$ cases
Fold 1 $(\mathbf{T}_1)$
Fold 2 $(\mathbf{T}_2)$
Fold K $(\mathbf{T}_K)$
losses using $\hat{f}^1$ for these cases are computed
fitting to these cases produces $\hat{f}^1$

Cross-validation

- More precisely, *K*-fold cross-validation proceeds by



1. randomly breaking the training set into K disjoint roughly equal-sized pieces ("**folds**"), say $\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_K$,
2. training on each of the reduced training sets $\boldsymbol{T} - \boldsymbol{T}_k$ (that we will call corresponding "**remainders**") to produce K predictors $\hat{f}^k$,
3. letting $k(i)$ be the index of the fold $\boldsymbol{T}_k$ containing training case i, and computing the cross-validation error

$$CV\left(\hat{f}\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(\hat{f}^{k(i)}\left(\boldsymbol{x}_i\right), y_i\right)$$

that one hopes approximates Err.

Cross-validation

- The case of K=N (all folds with a single case in them) has been called Leave One Out (LOO) cross-validation
  - for some special situations there are slick computational tricks that make it relatively fast
  - folklore has generally held it to have small bias for approximating Err, but large variance (over training sets) … this negative has recently been strongly challenged, making it arguably the most attractive choice (unless it’s computationally impossible)
- For K<N the cross-validation error (even for fixed training set) is random because of the randomness involved in splitting into folds … in light of this, it is common to average cross-validation errors from multiple splittings
- Where LOO cross-validation isn’t employed, “standard” choices of numbers of folds are K=5 and K=10

Cross-validation

- It is absolutely essential that cross-validation take account of **all** that is to be done in the production of predictions---it must be applied to the entire methodology used if one hopes to gain a reliable picture of likely prediction efficacy

- For example, if one is going to standardize input variables before fitting some kind of predictor, that standardization must be redone on each fold

- To put it another way, whatever one will do based on the entire training set in order to make predictions for new cases must be done separately (thus *K* times) on each "remainder" to build the predictions for the corresponding "fold" used to produce the cross-validation error

- Violation of this principle typically produces overly optimistic projections for method performance