Resampling methods

Fraida Fund

Contents

Resampling methods

Simple train/test split - basic idea

- Divide data into training and test sets
- For each candidate model order p, learn model parameters $\hat{\beta}$ on training set
- Measure error on test set
- Select model order p and corresponding $\hat{\beta}$ that minimizes error on test set

Simple train/test split

- Get data X, y
- Split into training X_{tr}, y_{tr} and test X_{ts}, y_{ts}
- Loop over models of increasing complexity: For $p = 1$ to p_{max},
  - **Fit**: $\hat{\beta}_p = fit_p(X_{tr}, y_{tr})$
  - **Predict**: $\hat{y}_{ts} = pred(X_{ts}, \hat{\beta}_p)$
  - **Score**: $S_p = score(y_{ts}, \hat{y}_{ts})$
- Select model order with best score (min loss/max perf): $\hat{p} = \text{argmin}_p S_p$

Problems with simple train/test split

- Fitted model, and test error, varies a lot depending on samples selected for training
- Fewer samples available for estimating parameters
- Especially bad for problems with small number of samples

Resampling

Resampling methods:

- Repeatedly draw samples from the training data.
- Re-fit the model on each new sample.
- Use all of the re-fitted models to draw conclusions

K-fold cross validation

Alternative to simple train/test split:

- Divide data into K equal-sized parts (typically 5, 10)
- Use $K - 1$ parts for training, last part for test
- Average over K test choices
- Gives better estimate of test error

K-fold CV illustrated

K-fold CV - pseudocode (1)

Outer loop over folds: for $i = 1$ to K

- Split into training X_{tr}, y_{tr} and test X_{ts}, y_{ts}
- **Inner loop** over models of increasing complexity: For $p = 1$ to p_{max},
  - **Fit**: $\hat{\beta}_p = fit_p(X_{tr}, y_{tr}$
  - **Predict**: $\hat{y}_{ts} = pred(X_{ts}, \hat{\beta}_p$
  - **Score**: $S_{p,i} = score(y_{ts}, \hat{y}_{ts})$

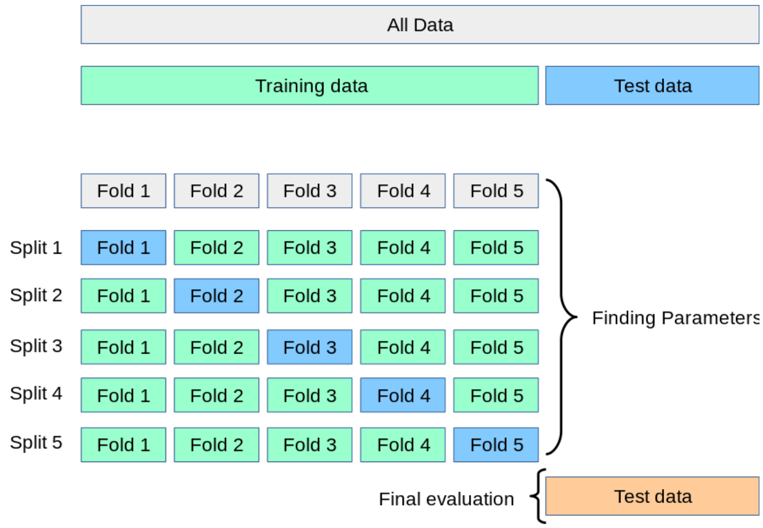

Figure 1: K-fold CV

K-fold CV - pseudocode (2)

- Find average score (across K scores) for each model: $\bar{S}_p$
- Select model with best *average* score: $\hat{p} = \text{argmin}_p \bar{S}_p$

K-fold CV - how to divide?

How to split?

- Avoid data leakage between parts.
- Stratified K-fold CV: make sure distribution of classes is similar in each part.

Leave one out cross validation (LOOCV)

- Let $K = N$
- One sample is left out on each iteration
- Often used when N is small

Bootstrapping

- Basic idea: Sampling **with replacement**
- Each "bootstrap training set" is *same size* as full training set, and is created by sampling with replacement
- Some samples will appear more than once, some samples not at all
- Bootstrap method will underestimate true prediction error

Using resampling methods

Two ways to use CV:

- Use CV to select "best" model; for each candidate model, evaluate CV error, and select model with least CV error
- When the "best" model is known, use CV to estimate test error

Bootstrapping is also used to estimate test error.

One standard error rule

- Model selection that minimizes mean error often results in too-complex model
- One standard error rule: use simplest model where mean error is within one SE of the minimum mean error

One standard error rule - algorithm (1)

- Given data X, y
- Compute score $S(p, i)$ for model p on fold i (of K)
- Compute average ($\bar{S}_p$), standard deviation σ_p, and standard error of scores:

$$SE_p = \frac{\sigma_p}{\sqrt{K-1}}$$

One standard error rule - algorithm (2)

- Normal rule for model selection:

$$\hat{p}_0 = \underset{p}{\text{argmin}}\, \bar{S}_p$$

- Compute target score: $S_t = \bar{S}_{p_0} + SE_{p_0}$
- **One SE rule**: select simplest model with score lower than target

$$\hat{p} = min\{p | \bar{S}_p \leq S_t\}$$

Final performance estimate with resampling

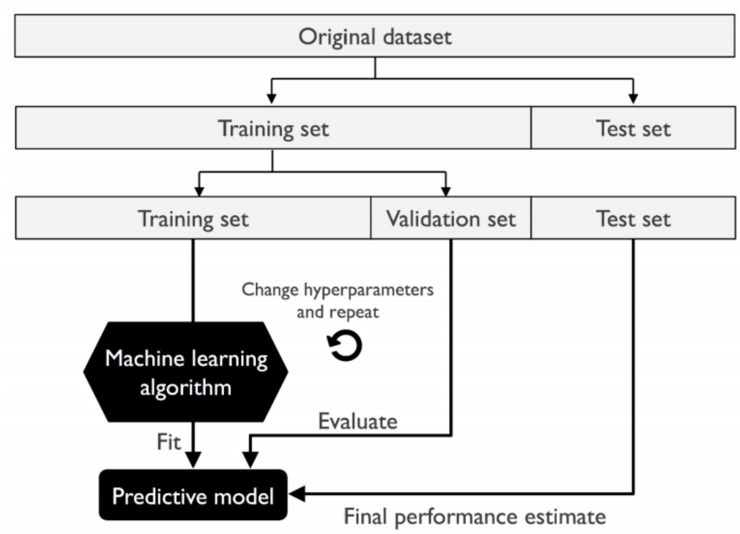

Figure 2: Hyperparameter tuning vs final performance evaluation