

# Machine Learning for Economists: Part 1 – Cross-Validation

Michal Andrle  
International Monetary Fund

Washington, D.C.,  
October, 2018

## Disclaimer #1:

**The views expressed herein are those of the authors and should not be attributed to the International Monetary Fund, its Executive Board, or its management.**

# MODEL ASSESSMENT AND SELECTION

# Model Assessment and Selection

How good is what you have? What model setup to pick?

## 1. Model Assessment and Tuning

- ▶ Optimizing **hyper-parameters** and architecture
- ▶ **Over fitting** assessment

## 2. Model Selection and Combination

- ▶ Model **selection**
- ▶ Model combination or **stacking** (super learners)

# Model Assessment and Selection

Key choices:

## 1. How the generalization error is estimated?

- ▶ True model and distributions unknown
- ▶ In-sample vs. out-of-sample tests
- ▶ cross validation

## 2. Criterion for model testing

- ▶ What is the right criterion?
- ▶ For example: least squares or absolute deviation?
- ▶ Decision utility, thresholds, ...
- ▶ ...

# Estimation of Model Error

How to estimate the model error using **only observed data**?

## 1. Training error (in-sample)

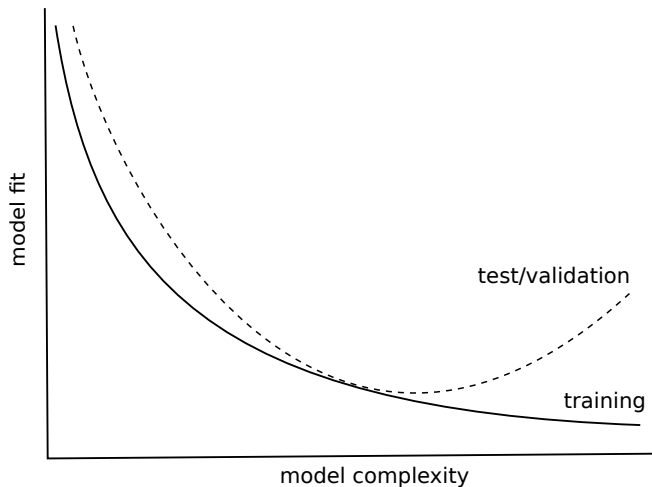
- ▶ In-sample fit over-optimistic
- ▶ Information criteria penalizing complexity (AIC, BIC, ...)

## 2. Generalization error (out-of-sample)

- ▶ Cross-validation (k-fold, LOOCV, ...)
- ▶ [Rolling] Windows estimation

**For flexible ML models training error can easily be zero!**

# Estimation of Model Error



Note: Optimizing hyper-parameters (complexity) over multiple dimension is common...

# Train, Test, Validate – When Data is Plentiful

## Partition the Data Set



### 1. Train

- ▶ Estimate the model(s) for a given value of hyper-parameters

### 2. Validate

- ▶ Optimize the hyper-parameters on a testing sample
- ▶ Find the weights for model stacking/combination

### 3. Test

- ▶ Final assessment of the model (ensemble) performance
- ▶ **No further tuning** or estimation using the test set



## Potential Issue with One Validation Set

The validation estimate of the test error can be highly variable

It is just one estimate of many potential split of the data into training and validation

# Cross-Validation

Split the data repeatedly into a training and validation set.

## 1. **K-fold Cross-Validation**

- + Simplest and most widely used
- + Available for all types of models
- More computations than for with the hold-out set

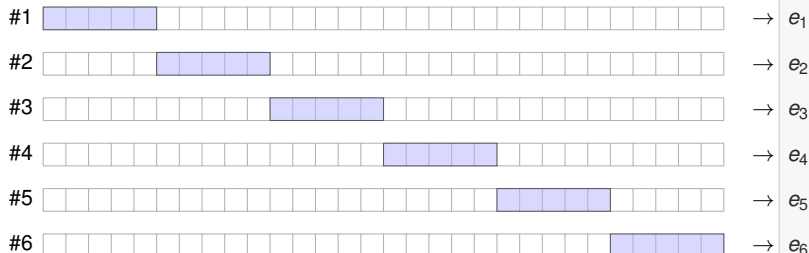
First papers on cross-validation is by Stone (1974) and Geisser (1975)...

# K-fold cross-validation IID data


Full data sample – resample all points (shuffle)



Train each model  $6 \times \dots$  [commonly,  $K=5$  or  $K=10$  but no rule...]



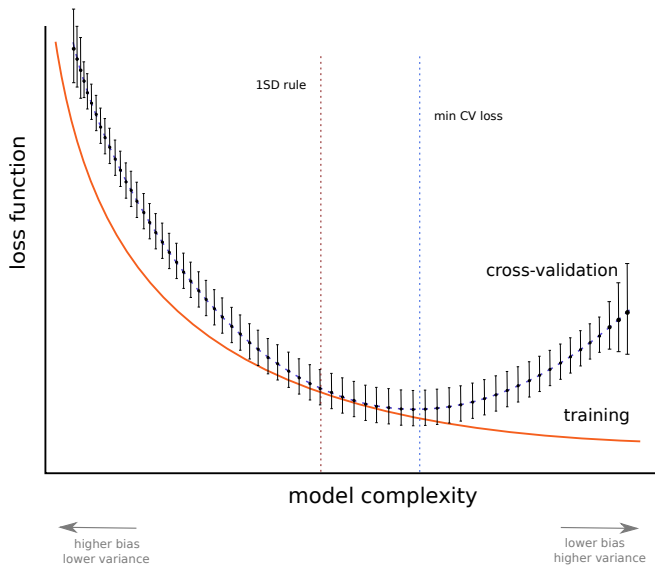
 training set

 test set

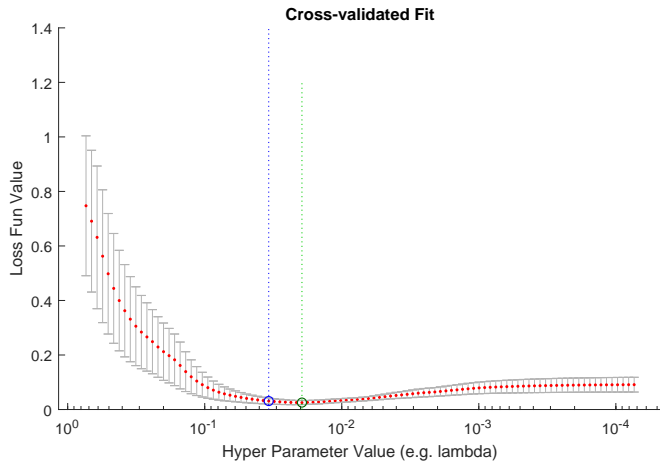
$mean(\mathbf{e})$   
 $std(\mathbf{e})$

Here  $e_k$  denotes the value of the loss/criterion function.

# Cross-Validation and Complexity Selection



# Cross-Validation and Complexity Selection



# K-fold cross-validation of iid data

K-fold CV is a 'fit and predict' cycle:

1. Divide the data  $\{1, \dots, N\}$  into  $K$  subsets (folds,  $F$ ),  $K \ll N$
2. For  $k = 1 : K$ 
  - 2.1 Train model on  $(y_i, x_i)$ ,  $i \notin F_k$ , test on  $(y_i, x_i)$ ,  $i \in F_k$
  - 2.2 For each value of tuning/hyper-parameter,  $\lambda \in \{\lambda_1, \dots, \lambda_m\}$

estimate the model  $\hat{f}_{-k}^\lambda$  on training set

$$e_k(\lambda) = \frac{1}{n} \sum_{i \in F_k} D(y_i - \hat{f}_{-k}^\lambda(x_i))$$

3. For each  $\lambda$  compute average (median, ...) across all folds:

$$CV(\lambda) = \frac{1}{K} \sum_{k=1}^K e_k(\lambda)$$

4. Also easy to compute 'standard error' of  $CV(\lambda)$ s:

$$SE_{cv}(\lambda) = \text{std}\{CV_1(\lambda), \dots, CV_K(\lambda)\} / \sqrt{K}$$

# Selecting Appropriate Complexity

- ▶ **Pick Optimal Value**

$$\hat{\lambda} = \arg \min_{\{\lambda_1, \dots, \lambda_m\}} CV(\lambda)$$

- ▶ **“One Standard Error Rule”**: Pick the Simpler Model

Choose model that is within one SE of the ‘optimal’ model

$$CV(\lambda) \leq CV(\hat{\lambda}) + SE(\hat{\lambda})$$

# Cross-Validation – Details

Carefully distinguish cross validation for:

## 1. **Independently-distributed data**

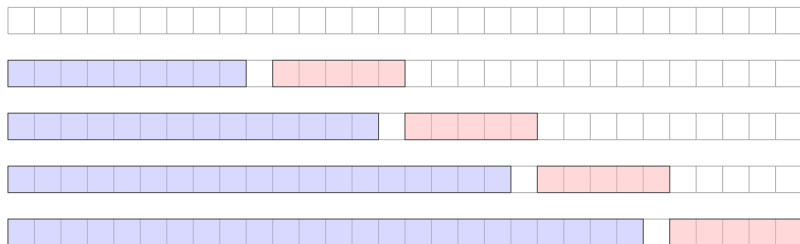
- ▶ Resample your data and divide into  $K$  folds
- ▶ Train model on  $K - 1$  folds, test on  $K$ -th fold

## 2. **Time series data** – more attention needed

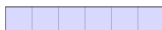
- ▶ Information leakage and dependencies may invalidate CV
- ▶ Cannot just resample the data as in the iid case
- ▶ [Rolling] Window estimation and testing
- ▶ h-Block CV
- ▶ Efficient Cross-validation of auto-regressive models is easy



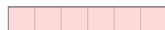
# Time-Series Out-of-Sample Validation – OoS



Unused data



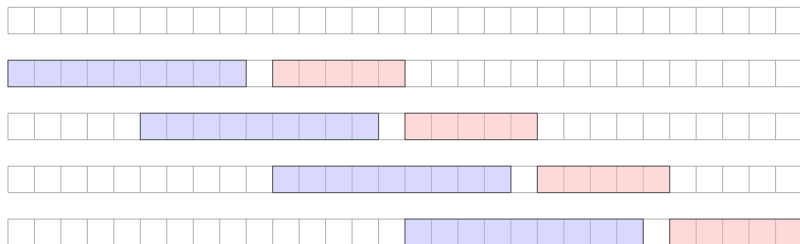
Training set



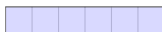
Validation set

Issue: Each training set features different # datapoints. . .

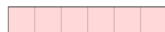
# Time-Series Rolling Out-of-Sample Validation – rOoS



Unused data



Training set



Validation set

rOoS can waste a lot of data...

# h-Block Cross Validation

For general models with dependent observations, Burman et al (1994) suggested **h-Block Cross Validation**

Before resampling,  $h$  observations preceding and following the current observation are left out. . .

The h-Block CV is very general but significantly reduces the effective sample size for CV

However, for auto-regressive models there is a better way. . .

# Cross-Validation of Auto-Regressive Models

For **auto-regressive** (AR) models cross-validation is available, contrary to common beliefs. . .

Standard K-fold cross-validation is valid for ARX(p) models, as long as the errors are uncorrelated. (Bergmeir et al. 2016)

## ARX(p) Cross-Validation: Intuition

Consider a non-linear ARX(p) model:

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-p}, x_t) + \varepsilon_t \quad (1)$$

$$\left[ \begin{array}{c|cccc} y_{21} & y_{20} & y_{19} & x_{21} \\ y_{20} & y_{19} & y_{18} & x_{20} \\ y_{19} & y_{18} & y_{17} & x_{19} \\ y_{18} & y_{17} & y_{16} & x_{18} \\ y_{17} & y_{16} & y_{15} & x_{17} \\ y_{16} & y_{15} & y_{14} & x_{16} \end{array} \right] \quad (2)$$

Note that the parameter estimates would be identical, if rows were permuted.

## ARX(p) Cross-Validation: Intuition

Consider a non-linear ARX(p) model:

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-p}, x_t) + \varepsilon_t$$

$$\left[ \begin{array}{c|ccc} y_{21} & y_{20} & y_{19} & x_{21} \\ y_{20} & y_{19} & y_{18} & x_{20} \\ y_{19} & y_{18} & y_{17} & x_{19} \\ y_{18} & y_{17} & y_{16} & x_{18} \\ y_{17} & y_{16} & y_{15} & x_{17} \\ y_{16} & y_{15} & y_{14} & x_{16} \end{array} \right] \quad (3)$$

Say, when  $y_{17}$  is missing, multiple rows,  $p$ , are affected. . .

## ARX(p) Cross-Validation: K-fold CV

Consider a non-linear ARX(p) model:

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-p}, x_t) + \varepsilon_t$$

$$\left[ \begin{array}{c|cccc} y_{21} & y_{20} & y_{19} & x_{21} \\ y_{20} & y_{19} & y_{18} & x_{20} \\ y_{19} & y_{18} & y_{17} & x_{19} \\ y_{18} & y_{17} & y_{16} & x_{18} \\ y_{17} & y_{16} & y_{15} & x_{17} \\ y_{16} & y_{15} & y_{14} & x_{16} \end{array} \right] \quad (4)$$

Bergmeir et al (2016) go further, showing that k-fold CV eliminating 'just' the whole rows of the stacked data is OK

# Stratified Cross-Validation

'Unbalanced datasets' have strongly unbalanced class proportions

e.g. of crises in the US data, % positive leukemia tests, ...

With unbalanced dataset a stratified cross-validation makes sure that each fold has roughly the same structure as the whole data set.



# Cross-Validation: Other Considerations

## ▶ Information leakage and wrong designs

- ▶ Not accounting properly for non-IID data
- ▶ Not shuffling the CV with IID data that is ordered in the dataset. . .
- ▶ Making choices (feature selection) based on the *whole* data set (incl. testing/hold out)
- ▶ Making variable transformation on the whole data set
- ▶ Not using 'grouped CV' for grouped data (say, panel data)
- ▶ . . .

## ▶ Using the test set multiple times for model development

- ▶ extreme hyperparameter tuning is as bad as 'p-hacking' and can overfit the test set
- ▶ CV "over-fitting" – testing too many hypotheses (A. Ng, 1997),
- ▶ CV is just 'combating' over-training/overfitting. . .

## So What About AIC, BIC, or $C_p$ ?

AIC, BIC, or  $C_p$  are **in-sample** prediction error estimates penalized for **model complexity**

AIC, BIC, or  $C_p$  penalize in-sample training error for effective number of parameters.

These simple criteria are usable for models **linear in parameters**, where **effective** number of parameters is defined. . .

Unadjusted in-sample training error estimate,  $err_{train}$ , is too optimistic an estimate of the true prediction error. . .

After all, complex models easily have  $err_{train} = 0$

## So What About AIC, BIC, or $C_p$ ?

For Akaike's Information Criterion (AIC) we have

$$\text{AIC} = \frac{2}{N} \log \text{lik} + 2 \frac{d}{N} \quad (5)$$

where  $d$  is **effective number of parameters**,  $N$  is sample size

For a model with hyper-parameters (tuning),  $\lambda$ , define

$$\text{AIC}(\lambda) = \text{err}_{\lambda, N} + 2 \frac{d(\lambda)}{N} \sigma_e^2, \quad (6)$$

and choose model and  $\lambda$  that minimizes  $\text{AIC}_\lambda$ .

AIC behaves like a cross-validation but is a crude tool and CV requires less assumptions. . .

**Thank you for your patience...**