Master in Financial Engineering (EPFL)
Financial Econometrics

Part II: Machine learning and Asset Pricing
Lecture 1: Key Machine Learning Concepts—a first-pass with supervised learning

Florian Pelgrin

EDHEC Business School

February - June 2019

1. Introduction

1. Introduction

Main objectives

- Discuss some key machine learning concepts in the context of supervised learning methods:
  - Issues with simple predictive methods (least squares and nearest neighbors);
  - Elements of statistical decision theory;
  - Bias-variance trade-off in machine learning;
  - Validation/resampling methods

2. A first application: Least squares and nearest neighbors

Overview

- Suppose that one has an outcome measurement (output feature, target) and wishes to predict it based on a set of input features (e.g., some explanatory variables)
- The training set of data is $\left\{\left(x_i^\top, y_i\right), i = 1, \cdots, n\right\}$.
- One implements two statistical methods:
  - ✓ Linear regression model;
  - ✓ Nearest neighbors.

- **Linear regression model**

$$y_i = \beta_0 + \sum_{i=1}^{p} x_{i,k}\beta_k + u_i$$

where p is the number of input features, u denotes the error term, $x_{i,k}$ is the k-th input feature for observation i, β_0 is the intercept (also known as the *bias* in machine learning),and $\beta_1 \cdots, \beta_p$ are the slope parameters.

The fitted value at the i-th input x_i is

$$\widehat{y}_i \equiv \widehat{y}(x_i) = x_i^\top \hat{\beta}$$

and, at any arbitrary input x_0, the prediction is:

$$\widehat{y}(x_0) = x_0^\top \hat{\beta}$$

Example: Training data on a pair of inputs and a response variable coded as zero (in blue) and 1 (in orange)

- Step 1: Fit the model
- Step 2: Define a classifier using the fit of the linear regression

$$Y^\star = \begin{cases} 1 & \text{if } \widehat{Y} > 0.5 \\ 0 & \text{otherwise.} \end{cases}$$

  The set $X^\top \widehat{\beta} = 0.5$ is a decision boundary.

- Step 3: Check for misclassification.

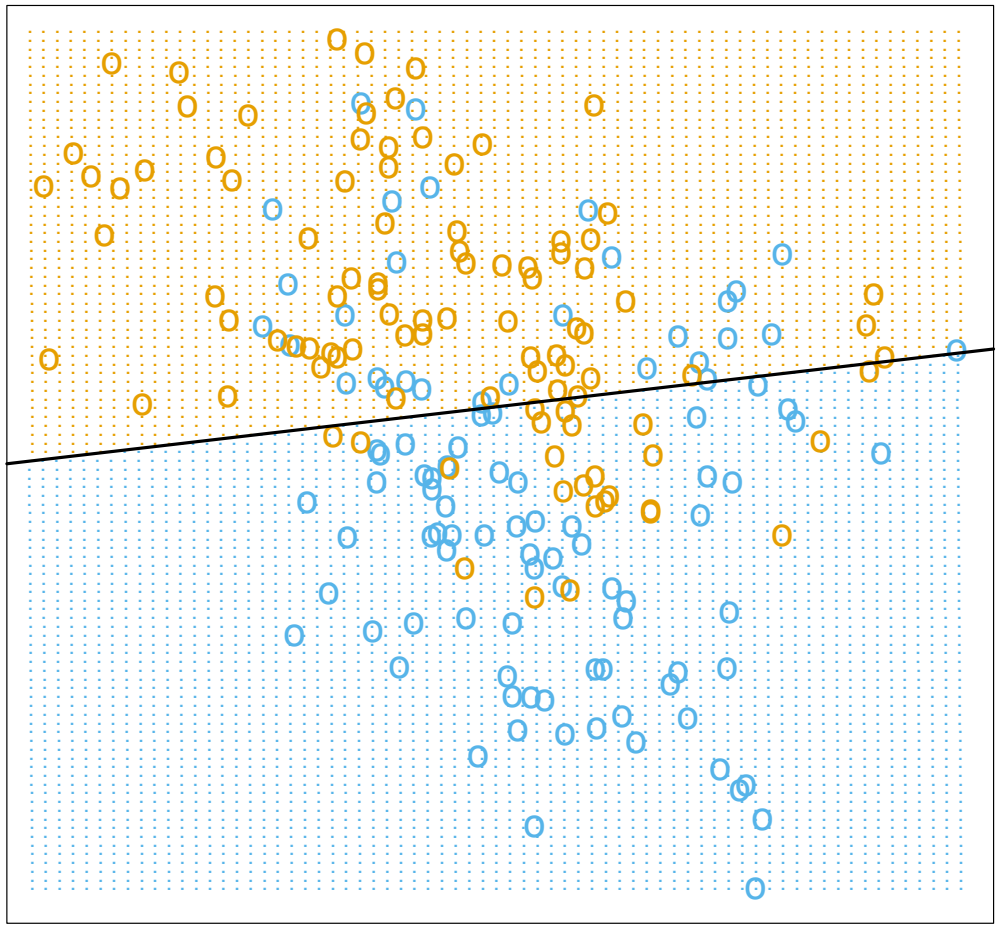

Figure: Linear regression of 0/1 response

Note: Orange (resp., blue) shaded region is the part of the input space classified as "1" (resp., "0"). The line is the decision boundary defined by $x^\top \widehat{\beta} = 0.5$.
Source: The Elements of statistical learning, Hastie et al. (2001).

Nearest neighbors

- **Nearest neighbors**: Make use of those observations (in the training set) closest in input space x to form the prediction

$$\widehat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i.$$

where $N_k(x)$ is the neighborhood of x defined by the k closest points x_i in the training sample.

Example: $\widehat{Y}$ is the proportion of orange circles in the neighborhood and it is assigned the value 1 if a majority of neighbors are orange circles.

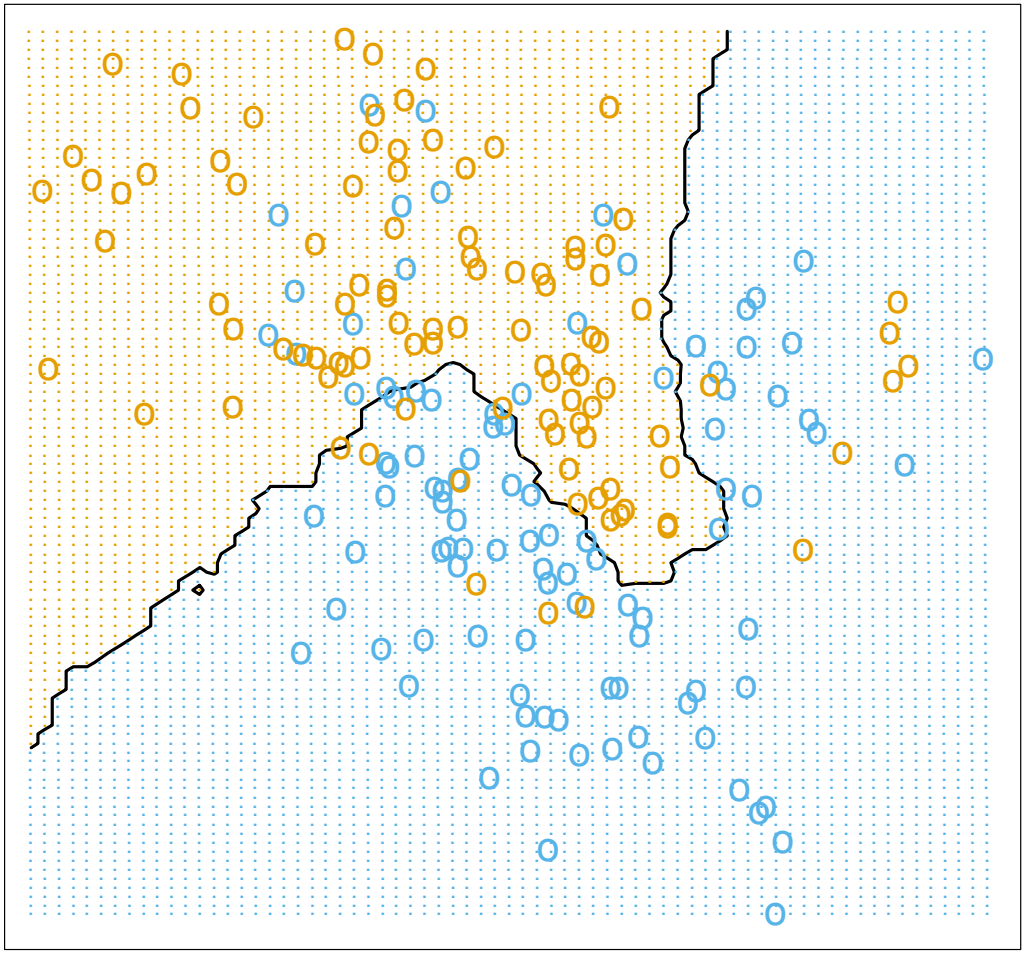

Figure: 15-nearest neighbor averaging

Note: Blue (resp., orange) points, $Y^\star$ correspond to 0 (resp., 1). The class is chosen by majority vote amongst the 15-nearest neighbors.
Source: The Elements of statistical learning, Hastie et al. (2001).

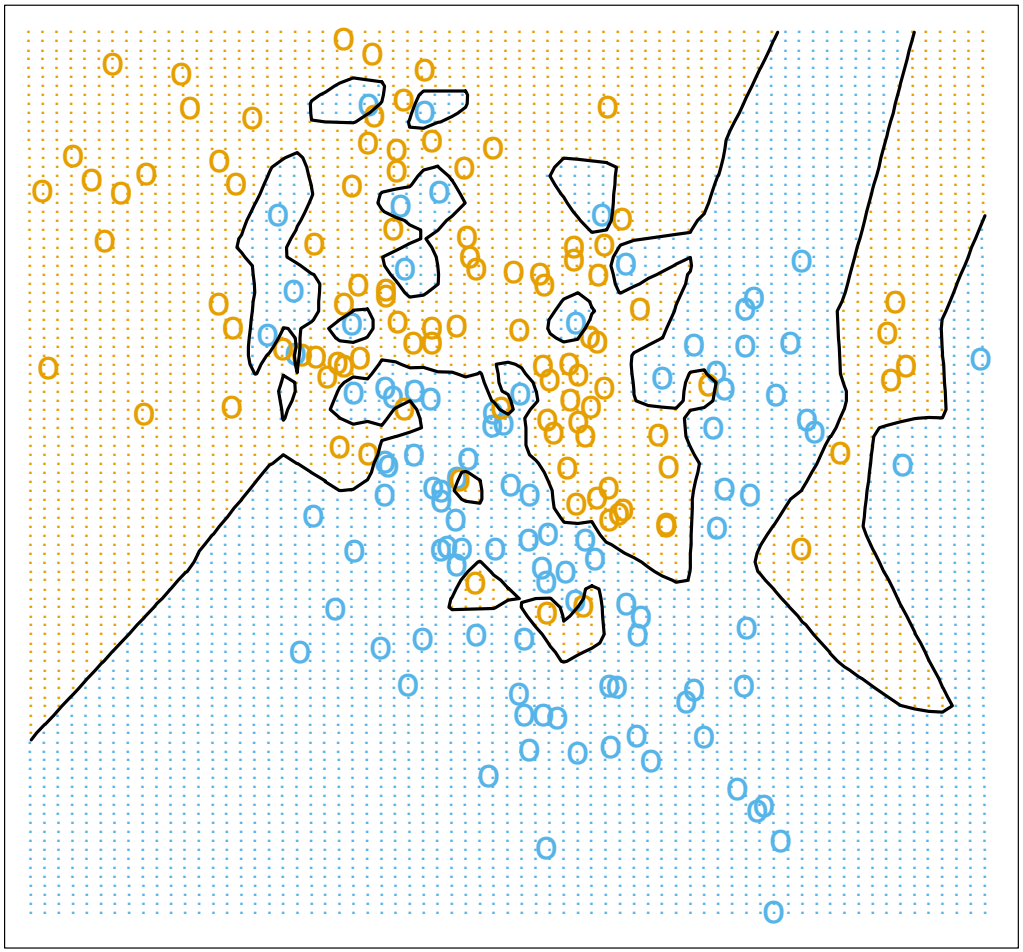

Figure: 1-nearest neighbor averaging

Source: The Elements of statistical learning, Hastie et al. (2001).

Insights

- Number of parameters: k *versus* $p \cdots$ or $\cdots$ n/k *versus* p
  - Flexibility (complexity)
  - Interpretability
  - High-dimensional problem
- Cannot use the sum-of-squared errors for k-N, otherwise always choose $k = 1$: Which criteria?
- Bias/Variance trade-off

- Performances of the linear model does depend on the appropriateness/effectivness of the linear decision boundary:
  - ✓ "Low" variance;
  - ✓ Potentially "high" bias.
- Performances of the k-nearest neighbors do not rely (at this stage) on any stringent assumptions but it depends on a handful of input points and their position:
  - ✓ "High" variance;
  - ✓ "Small" bias.
- Most of the recent procedures make use of these two simple procedures (kernel methods, local regressions, projection pursuit, neural network models).

3. Statistical decision theory

- Let $X \in \mathbb{R}^p$ denote a real valued random input vector and $Y \in \mathbb{R}$ a real valued output variable with joint distribution $\texttt{Pr}(X, Y)$.

- One seeks a **function** $f(X)$ for predicting Y taking values of the input $X \Rightarrow$ this requires a **loss function**, say $L(Y, f(X))$, e.g. the squared error loss

$$L(Y, f(X)) = (Y - f(X))^2,$$

- ...and thus a **criterion** for choosing f, say the Expected (Squared) Error Prediction

$$\begin{aligned} \texttt{EPE}(f) &= \mathbb{E}\left[(Y - f(X))^2\right] \\ &= \mathbb{E}_X \mathbb{E}_{Y|X}\left(\left[Y - f(X)\right]^2 | X\right) \end{aligned}$$

- By minimizing (pointwise) the EPE,

$$f(X) = \mathbb{E}\left[Y | X = x\right]$$

the solution is the conditional expectation (regression function).

- **k-nearest neighbors**:

$$\widehat{f}(x) = \texttt{Ave}\left(y_i | x_i \in N_k(x)\right)$$

where the conditional expectation is "replaced" by the sample "average" and the conditioning set includes the information "close" to the target point.

- **Least squares solution**:

$$\widehat{f}(x) = x^\top \widehat{\beta}$$

where $\widehat{\beta}$ is the empirical counterpart of $\left[\mathbb{E}\left(XX^\top\right)\right]^{-1}\mathbb{E}\left(XY\right)$.

- Conditional expectations are approximated by averages in both cases but model assumptions are different!
  - `NN`: Approximation by a locally constant function;
  - `LS`: Approximation by a globally linear function.

"Degrees of freedom":

- Specification of f (function approximation):
  - ✓ Linear/nonlinear specification
  - ✓ Kernel function and local regression,
  - ✓ Basis functions and dictionary methods, etc
- Specification of the criterion
  - ✓ Other norms, e.g., L_2 versus L_1

Norm	Loss function	Solution
L_1	$\mathbb{E}\left\|Y - f(X)\right\|$	Conditional median
L_2	$\mathbb{E}\left[(Y - f(X))^2\right]$	Conditional mean

  - ✓ Penalization and regularization

Norm	Empirical Loss function	Penalty
L_2	$\sum_{i=1}^{n} (y_i - f(x_i))^2$	$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda C(f)$

- Type of output: Quantitative *versus* qualitative data.

Examples

- **Kernel methods and local regression**

$$\sum_{i=1}^{n} K\lambda(x_0, x_i)\left(y_i - f_\theta(x_i)\right)^2$$

where $K\lambda$ is the kernel (weight function based on a given pdf) and λ denotes some hyperparameters (e.g., to control the width of the neighborhood).

Some interesting cases:

- ✓ $f\theta(x) = \theta_0$ (constant function): Nadaraya-Watson weighted average;
- ✓ $f\theta(x) = \theta_0 + \theta_1 x$: local linear regression model;
- ✓ $K_k(x, x_0) = I\left(\|x - x_0\| \leq \|x_{(k)} - x_0\|\right)$ where $x_{(k)}$ is the k-th ranked observation in distance from x_0 and $I(S)$ is the indicator of the set S: Nearest neighbor methods.

Examples (cont'd)

- **Basis functions**

$$f_\theta(x) = \sum_{m=1}^{M} \theta_m h_m(x)$$

where each h_m is a function of the input x:

- ✓ Spline basis functions;
- ✓ Radial basis functions;
- ✓ Single-layer feed-forward neural network model (adaptive basis function)...
- ✓ Possibly infinite set of candidate basis functions (known as *dictionary methods*).

- **Penalized regression**: Cubic smoothing spline is the solution to the penalized `LS` regression

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int \left[f''(x)\right]^2 dx$$

- In general, all of the models considered in machine learning have (at least) a **smoothing** or a **complexity parameter** (e.g., Lagrange multiplier of the penalty term, the width of the kernel, number of basis functions, etc).

- For instance, the local degree-m polynomial model ranges between a global polynomial (when the size of the window is infinitely large) and an interpolating fit (constant function) when the window size shrinks towards zero.

- These "hyperparameters" are critical!

Method	"Hyperparameters"
Global/parametric predictors	
Linear regression	Subset selection $\|\beta\|_0 = \sum_{j=1}^{p} 1_{\beta_j \neq 0}$
and generalizations	Lasso $\|\beta\|_1 = \sum_{j=1}^{p} \|\beta_j\|$
	Ridge $\|\beta\|_2 = \sum_{j=1}^{p} \beta_j^2$
	Elastic net: Combination of Lasso and ridge
Global/parametric predictors	
Non-linear regression	Number of polynomials, radial basis
Local/nonparametric predictors	
Decision/regression trees	Depth, number of nodes, minimal leaf size
Random forest	Number of trees, number of variables in each tree
	Complexity of each tree
Nearest neighbors	Number of neighbors
Kernel regression	Kernel bandwidth
Mixed predictors	
Deep learning	Number of levels (layers), number of neurons
(including ANN)	Connectivity of neurons
Splines	Number of knots

4. Bias-Variance trade-off

Overview

- What is the "variance" and the "bias" of a statistical learning method?
  - ✓ "Bias": Error introduced by approximating the "true" (or real-life) problem by a much simpler model;
  - ✓ "Variance": How much $\widehat{f}$ would change when considering different training data sets?

    "High variance" means that small changes in the training data set lead to large changes in $\widehat{f}$.

Expected test MSE

- Suppose that data arise from a model $Y = f(X) + u$ with $\mathbb{E}(u) = 0$ and $\mathbb{V}(u) = \sigma^2$.

- The **prediction**, **test**, **generalization error** or **expected test MSE** at x_0 is defined to be:

$$\begin{aligned} \texttt{EPE}(x_0) &= \mathbb{E}\left[\left(Y - \widehat{f}(x_0)\right)^2 | X = x_0\right] \\ &= \left(\mathbb{E}\left[\widehat{f}(x_0)\right] - f(x_0)\right)^2 + \mathbb{E}\left[\left(\widehat{f}(x_0) - \mathbb{E}\left[\widehat{f}(x_0)\right]\right)^2\right] + \sigma^2 \\ &= \text{Mean Squared error of } \widehat{f}(x_0) + \text{irreducible error} \end{aligned}$$

where the MSE is decomposed in two terms:

  - Bias component: "Squared bias" that can be improved using a more appropriate function;
  - Variance component: Variance of the predictions over the training/test set (that will most likely increase with the complexity of the model).

- **Example**: k-nearest neighbor regression

$$\begin{aligned}\texttt{EPE}(x_0) &= \texttt{Bias}^2(\widehat{f}_k(x_0)) + \mathbb{V}_{\mathcal{T}}(\widehat{f}_k(x_0)) + \sigma^2 \\ &= \left[f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})\right]^2 + \frac{\sigma^2}{k} + \sigma^2\end{aligned}$$

where the sequence of k-nearest neighbors to x_0 is denoted $(x_{(\ell)}, \ell = 1, \cdots, k)$.

Note that

- ✓ When k is small, the k closest neighbors are expected to have value $f_{x_{(\ell)}}$ close to $f(x_0)$;
- ✓ When k is large, the neighbors are further away...the bias might increase but the variance decreases as the inverse of k.

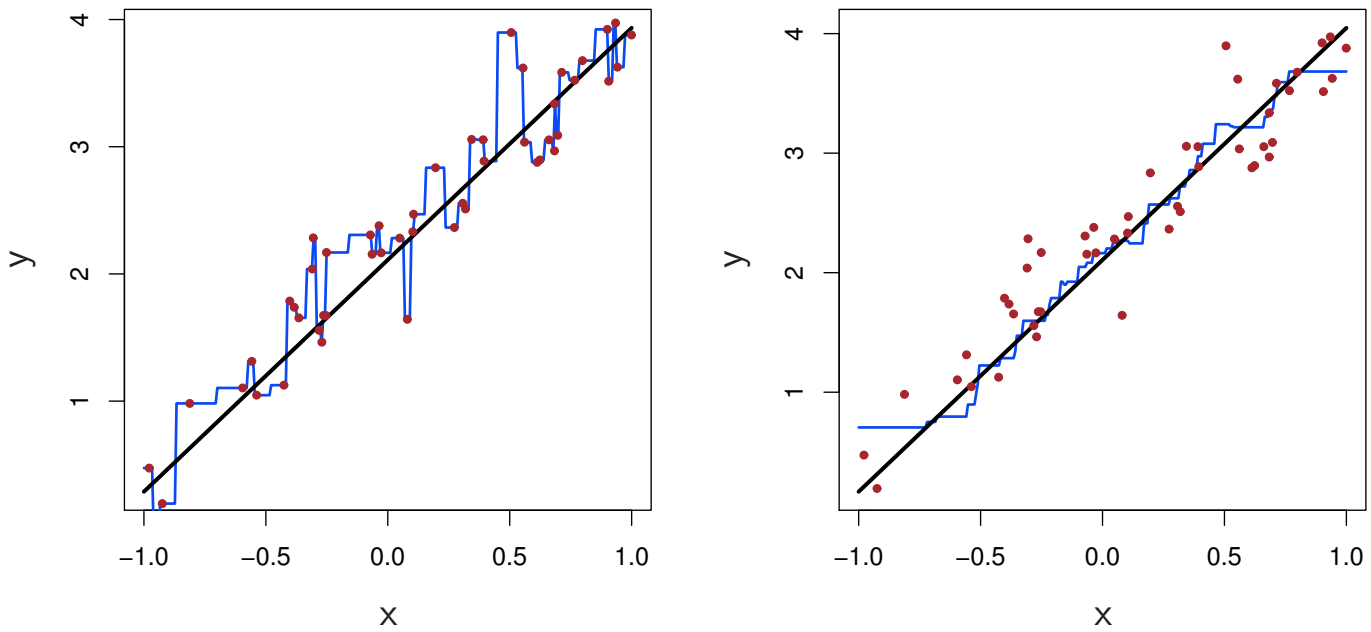

Figure: Nearest neighbors and linear regression

Note: The true relationship is linear (black solid line). Left panel corresponds to the one-nearest neighbor (after interpolating). Right panel represents the 9-nearest neighbors (smoother fit).

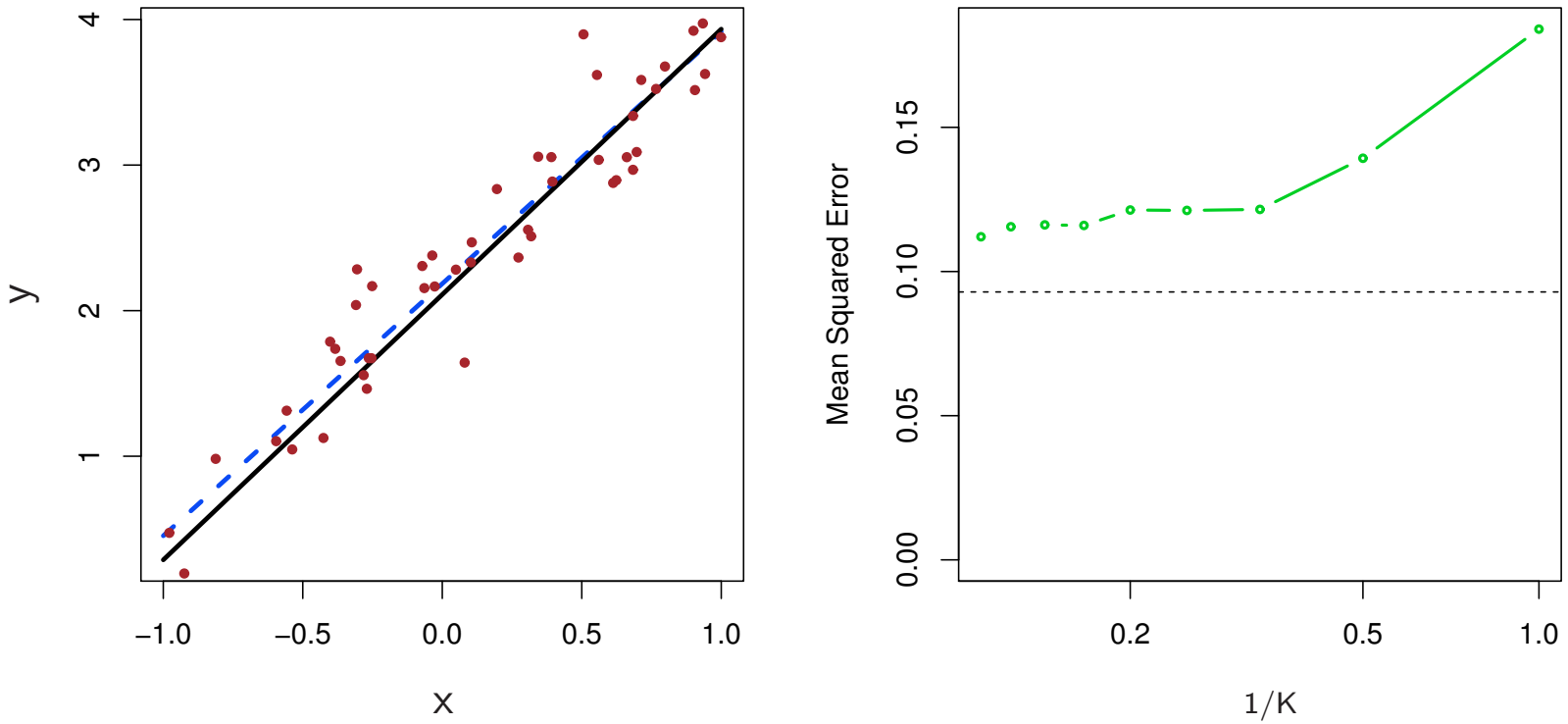

Figure: Nearest neighbors and linear regression

Note: The true relationship is linear (black solid line). Right panel represents the MSE of the LS model (dashed horizontal line), and the MSE of the nearest-neighbor model as a function of 1/k (on the log scale).

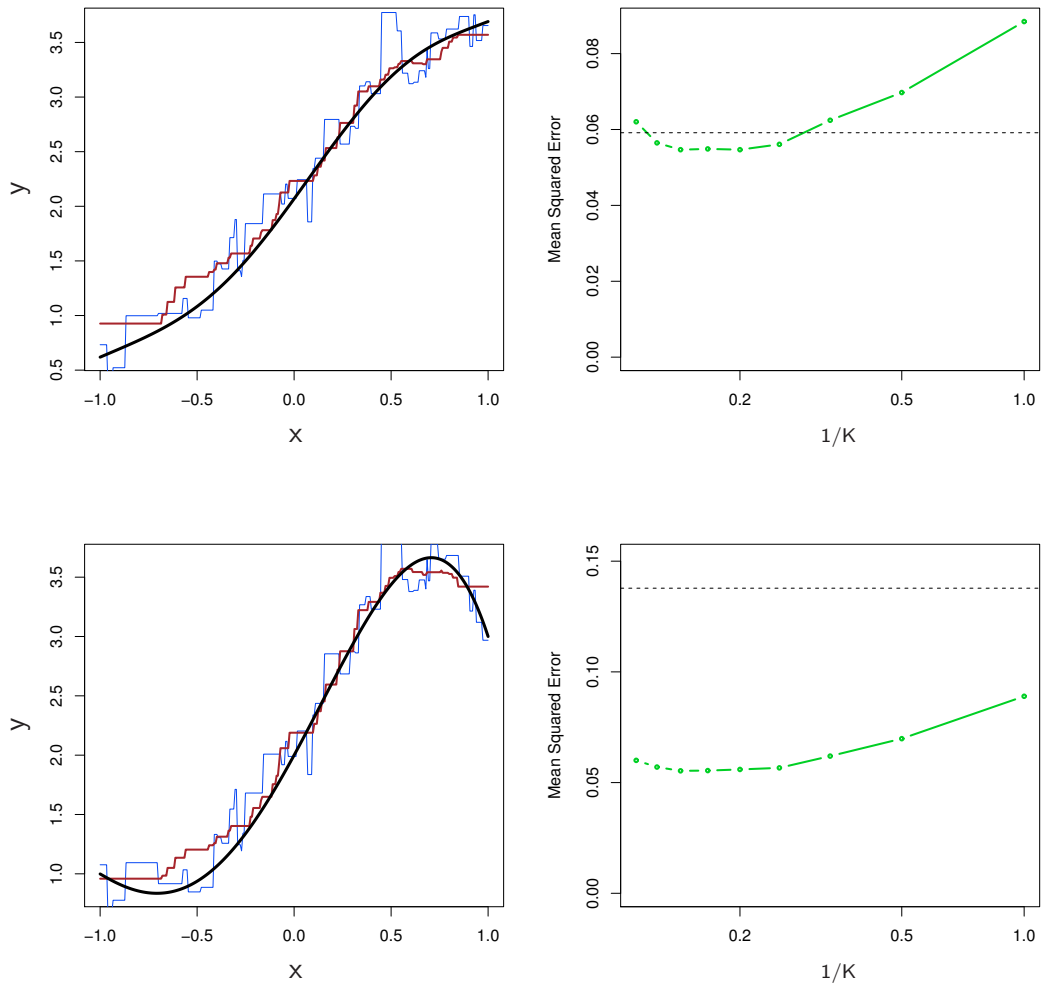

Figure: Nearest neighbors and linear regression

Note: The true relationship is slightly non-linear (black solid line). Left panel represent the NN with k = 1 (blue) and k = 9 (red). Right panel represents the MSE of the LS model (dashed horizontal line), and the MSE of the nearest-neighbor model is a function of 1/k (on the log scale).

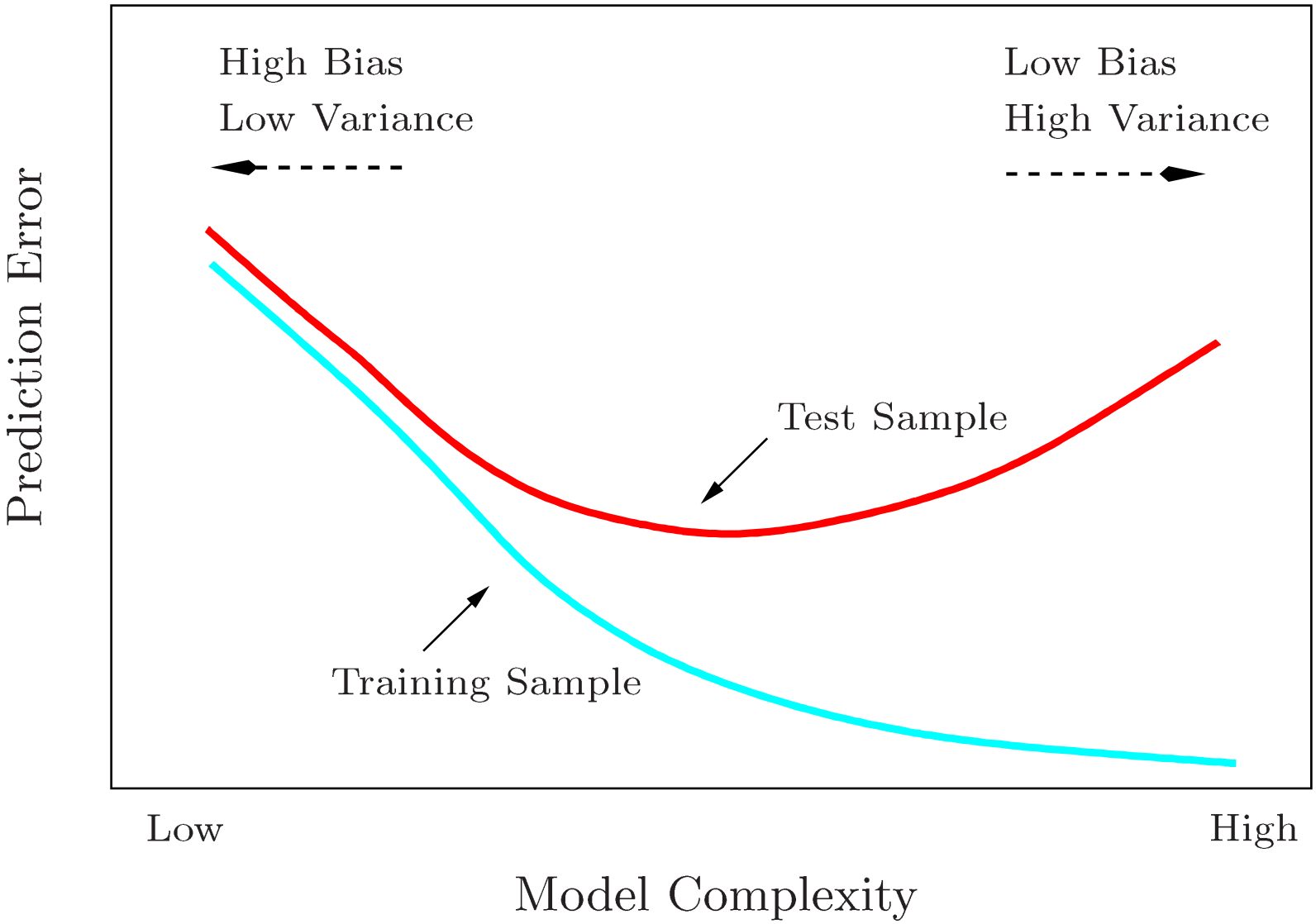

Figure: Fundamental relationship between prediction error and model complexity

Some insights

- The training error generally decreases w.r.t. model complexity (i.e., data is fit harder and harder);
- But too much fitting:
  - ✓ Model is adapted for training data but generalizes poorly (test sample);
  - ✓ Predictions have large variance when model complexity increases.
- The "test error" can be decomposed into the MSE and the incompressible error.
- There is an "optimal" cut-off for model complexity using test sample...but which one?

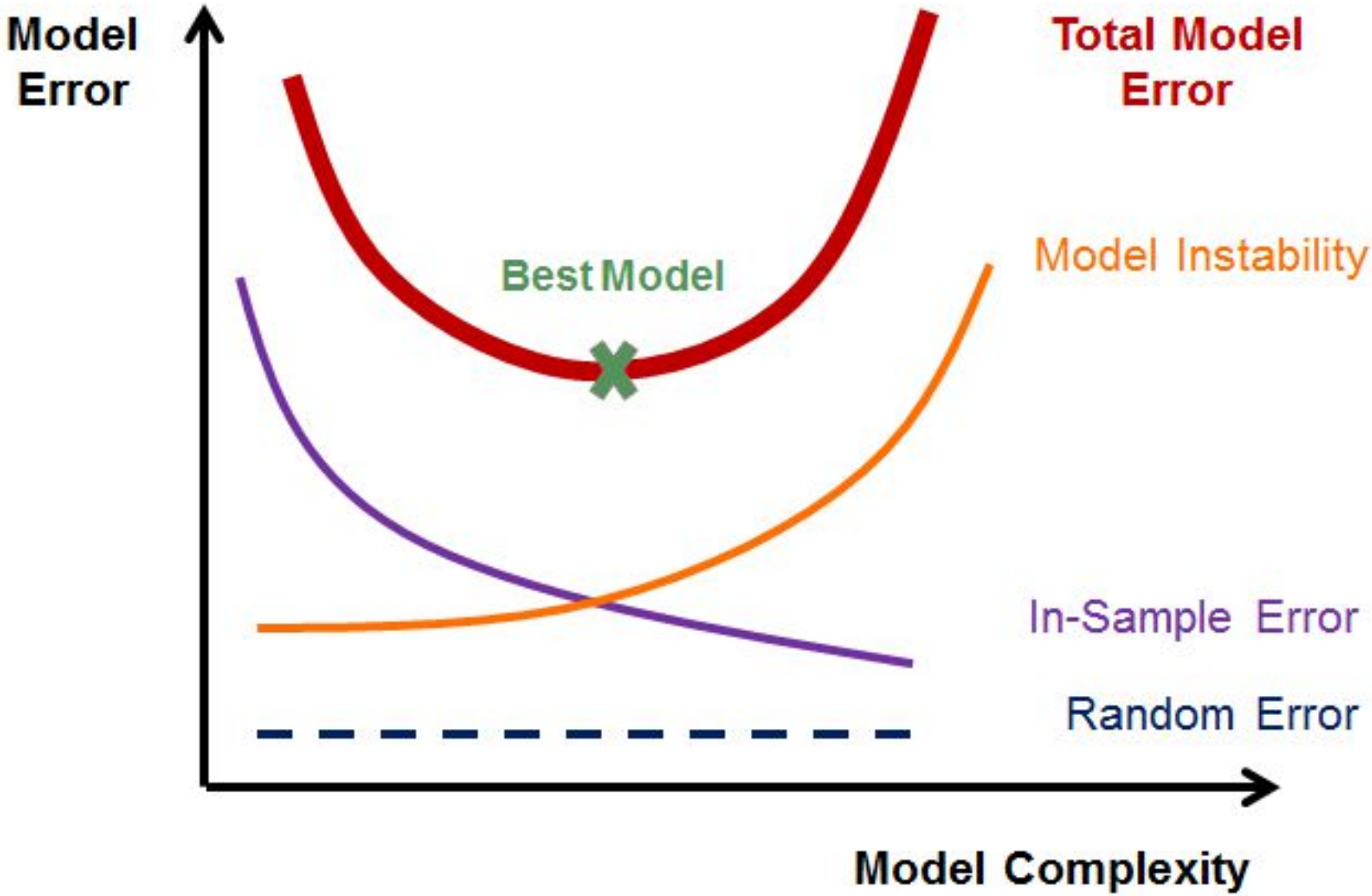

Figure: Existence of a "best" model?

Application:

- Perform some simulations with different true "f" (black curve);
- Fit more or less noisy data with two methods:
  - ✓ Linear regression;
  - ✓ Smoothing splines (with two different levels of smoothness).
- Compute the training and test MSE:
  - ✓ Orange squares: linear regression;
  - ✓ Blue squares: smoothing splines (less flexibility);
  - ✓ Green squares: smoothing splines (more flexibility).

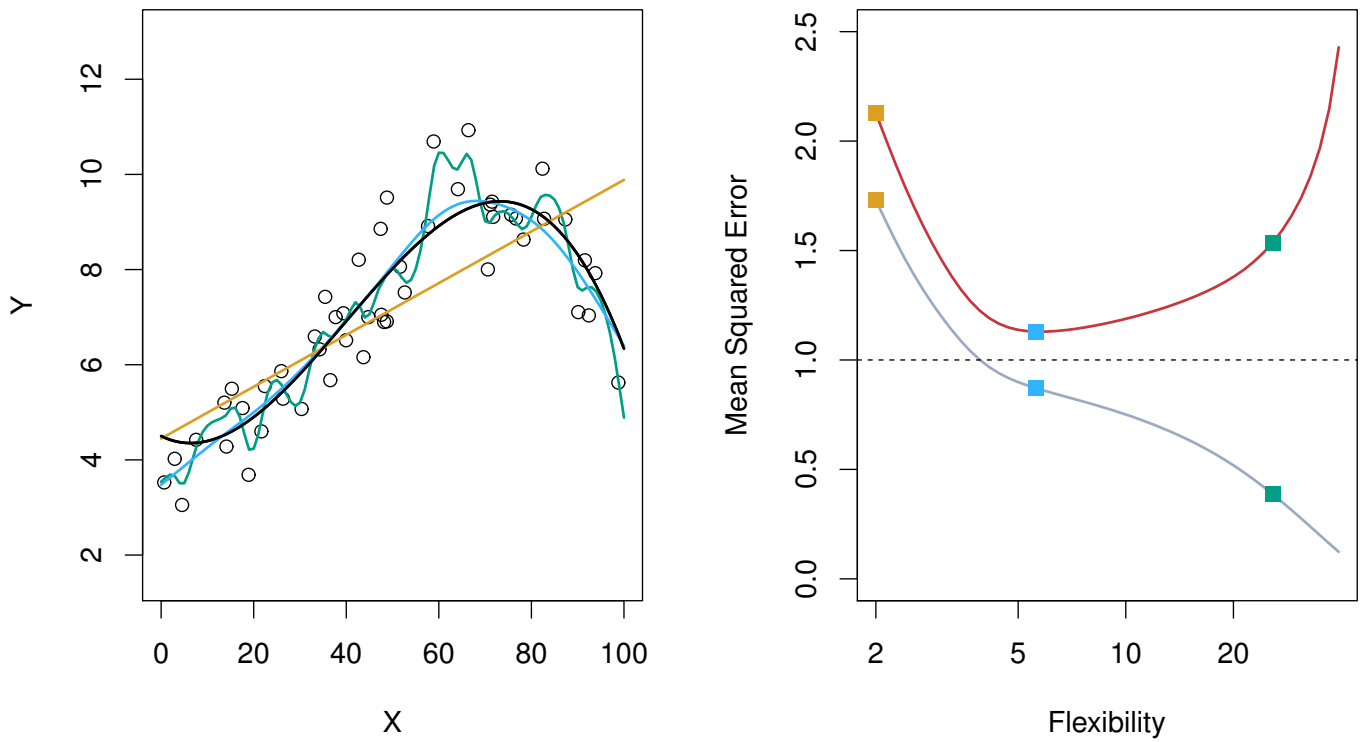

Figure: Noisy training data and complex function

Note: Left panel represents "true model" (black curve).Circles represents noisy training data. The right panel represents the training MSE (grey curve) and the test MSE (red curve). Squares are the training and test MSE for the three fits on the left-hand panel.

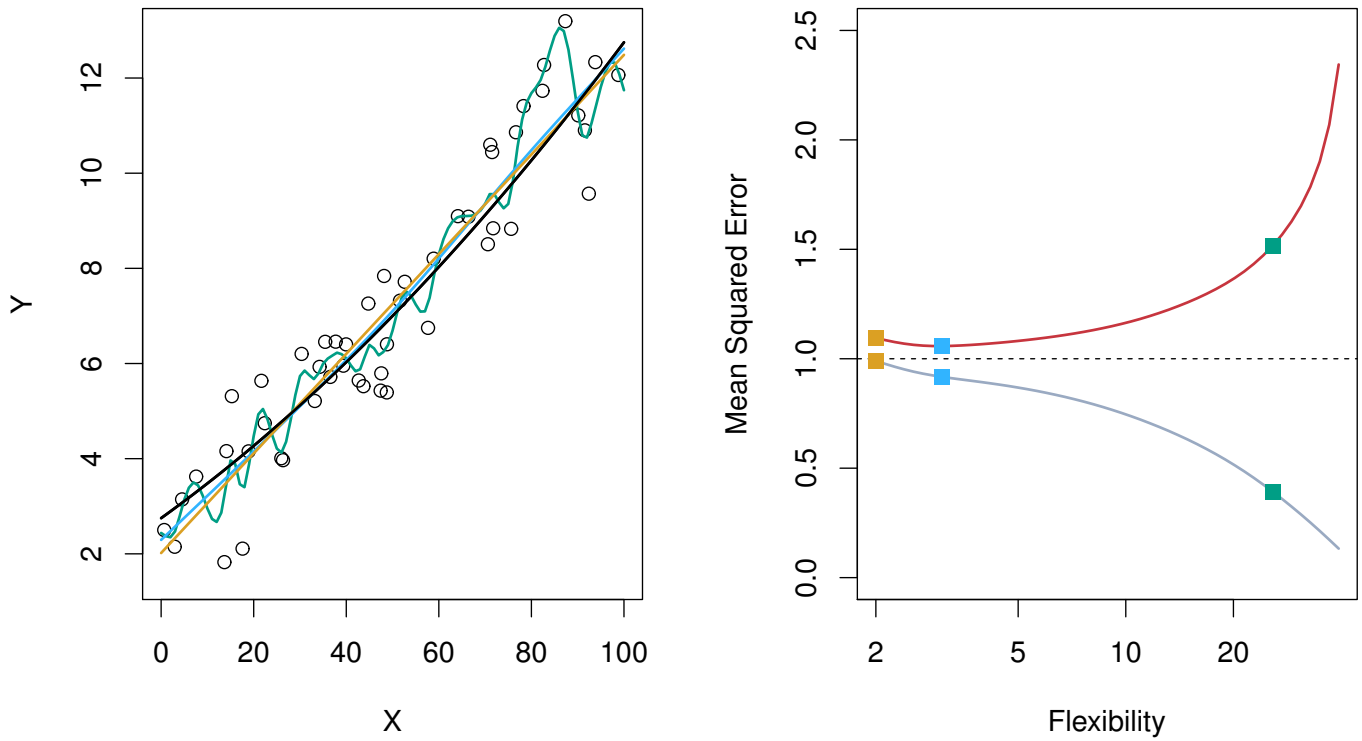

Figure: Linear-ish function

Note: The black curve is the "true model" and is roughly linear. Test MSE falls but then rises showing linear model is best.

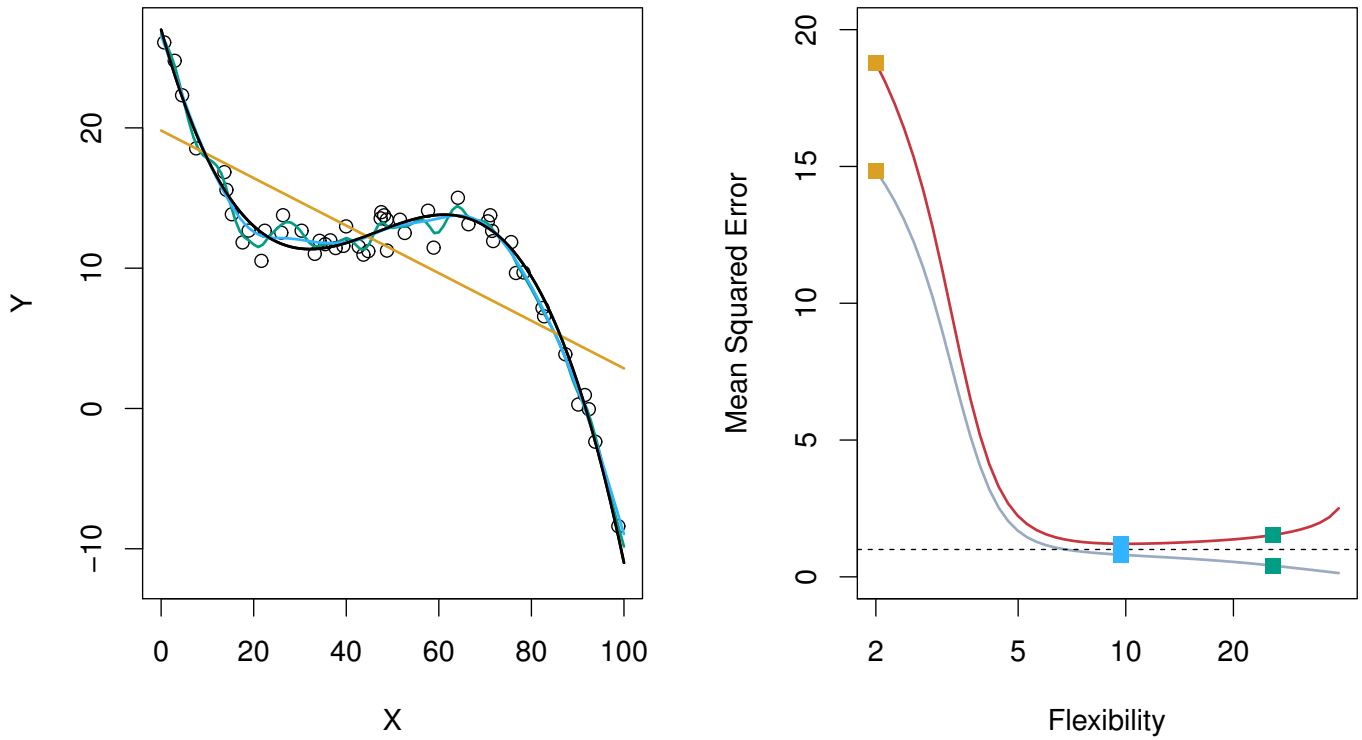

Figure: Highly non-linear function and "smooth" data

Note: The black curve is the "true model" and is highly non-linear, and data is not too noisy. Test and training MSE fall quickly as complexity increases before test MSE starts increasing slowly..

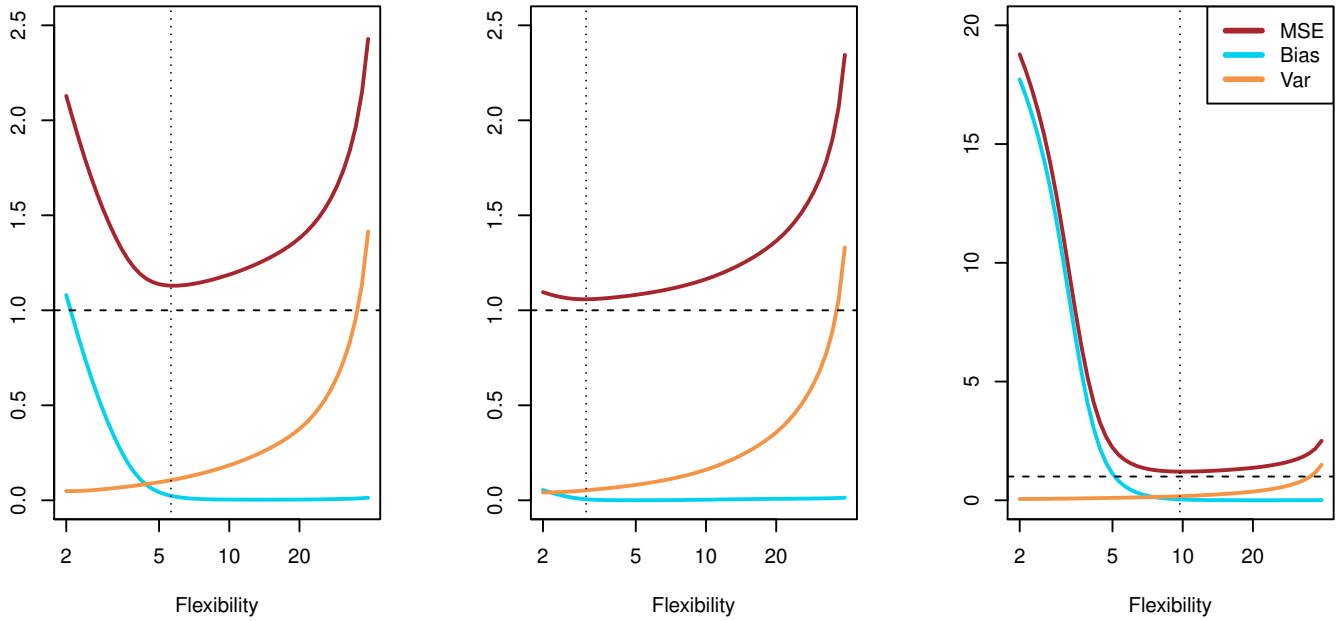

Figure: Decomposition of the bias-variance trade-off

Some insights

- A **large sample size** and a **small number of predictors**: flexibility is better since it allows to take full advantage of the large sample size;
- A **small sample size** and a **large number of predictors**: flexibility will generally cause overfitting due to the small sample size;
- A **highly non-linear relationship** between the predictors and the output: flexibility is generally necessary to capture the non-linear effect;
- A **large variance of the error term**: flexibility is generally worse since one fits too much of the noise in the problem;

Some insights (cont'd)

- In practise, f is unobserved and thus it is generally not possible to compute the test MSE, bias, or variance for a statistical learning method. Statistical learning methods can be extremely flexible and thus eliminate the bias but it does not guarantee that they will outperform a much simple method (e.g., the linear regression model).

- One way to evaluate the test MSE is to make use of resampling methods:

  - ✓ Validation set
  - ✓ Cross-validation
  - ✓ Bootstrapping, etc.

- Complexity (flexibility) and interpretability

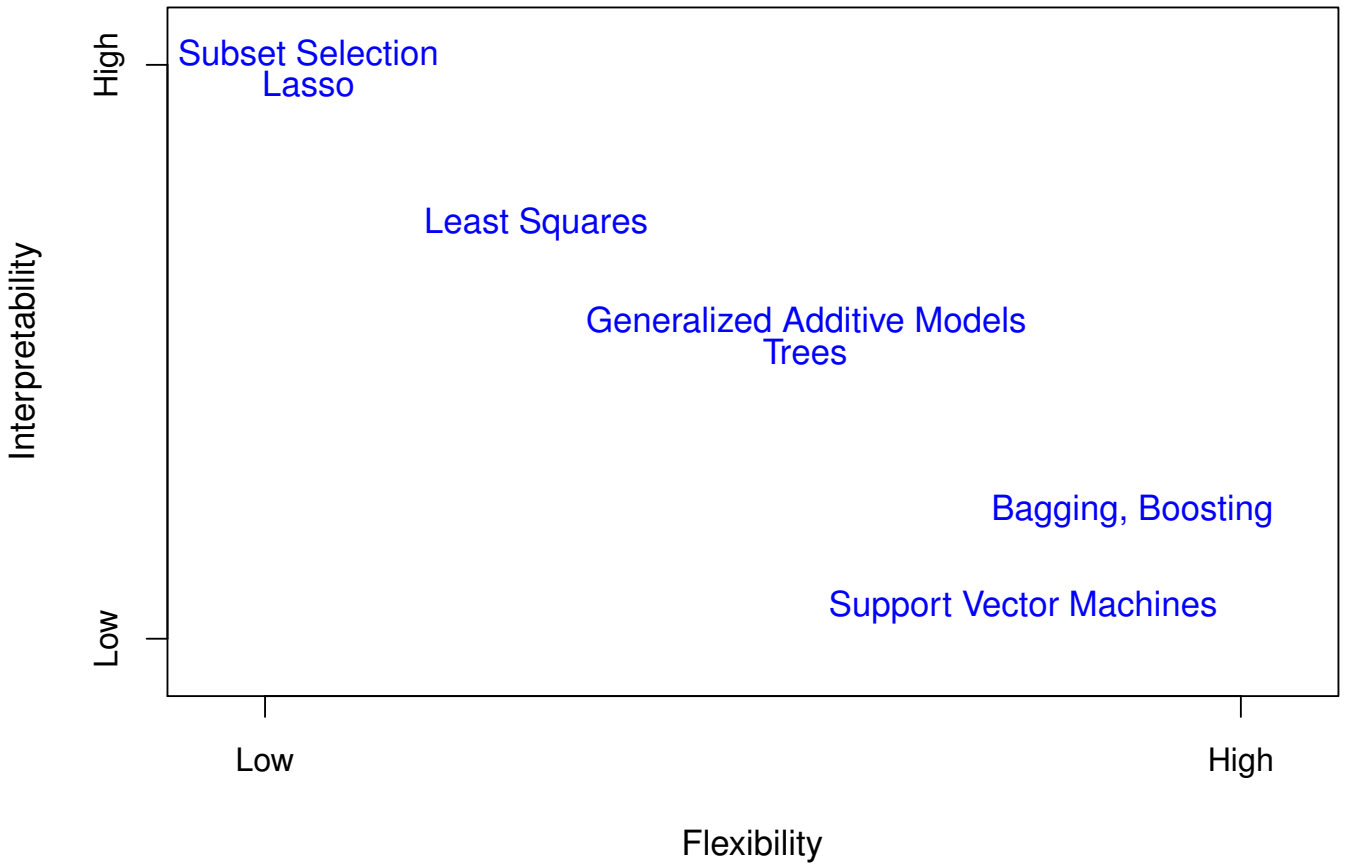

Figure: Complexity and interpretability

5. Resampling methods

Overview

- Main idea of resampling methods: Draw samples repeatedly from a training set and refit the model of interest on each sample in order to obtain further/additional information about the fitted model (on the training sample).

- Among others,

  - ✓ Cross-validation
  - ✓ Bootstrap (same as in econometrics!)

- For instance, cross-validation can be used (i) to estimate the test error (associated with a given statistical learning method)—**model assessment**, (ii) to select the appropriate level of flexibility—**model selection**.

The validation set approach

A first approach that proceeds as follows:

1. Divide randomly the set of observations into two parts:
   - ✓ Training set
   - ✓ Validation set or hold-out set
2. Fit the model on the training set and predict the responses for the observations in the validation set—the corresponding *validation set error rate* provides an *estimate of the test error rate*.
3. Repeat Step 1 and Step 2.

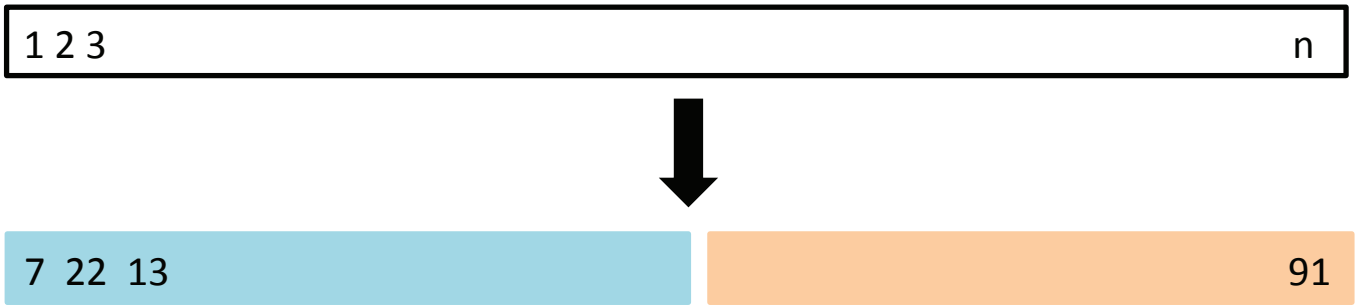

Figure: Validation set approach

Note: A set of n observations is randomly split into a **training set** (in blue) and a **validation set** (in beige)—the statistical learning method is fit on the training set and its performances are evaluated on the validation set.

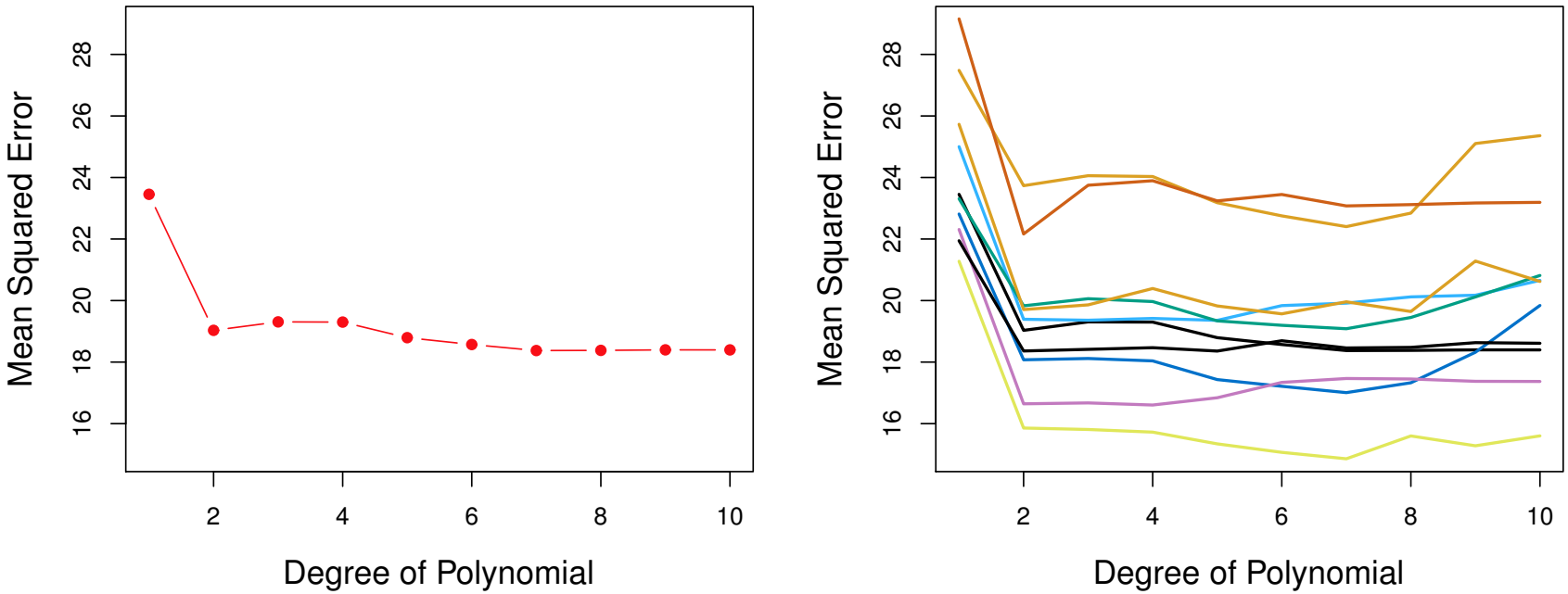

Figure: Implementation of a validation set approach

Note: Left panels correspond to the validation error estimates for a single split into training and validation data sets. Right panels corresponds to a 10-repetition.

Main insights:

- The validation set approach is conceptually simple and is easy to implement (especially for cross-section data);
- But,
  - ✓ The validation estimate of the test error rate can be highly variable: it does depends on the partitioning between the training set and the validation set, and especially the observations included in these sets!
  - ✓ Only a subset of the observations is used to train the model. In general, statistical models tend to perform worse in the presence of fewer observations—this suggests that the validation set error rate might tend to overestimate the test error rate (on the entire data set).
- This leads to cross-validation methods.

Cross-validation methods: Leave-One-Out Cross-Validation (LOOCV)

1. Divide the set of observations into two parts:
   - ✓ Training set: $\left\{\left(x_i^\top, y_i\right), i = 2, \cdots, n\right\}$
   - ✓ Validation set or hold-out set: $\left(x_1^\top, y_1\right)$
2. Fit the model on the training set (of dimension $n-1$) and predict the responses for the observations in the validation set (of dimension one)—$\left(y_1 - \widehat{y}_1\right)^2$ (empirical mean squared error for the first error) provides an approximately unbiased estimate for the test error.
3. Repeat Step 1 and Step 2 by selecting $\left(x_2^\top, y_2\right)$ for the validation set, and then $\left(x_3^\top, y_3\right)$, etc.

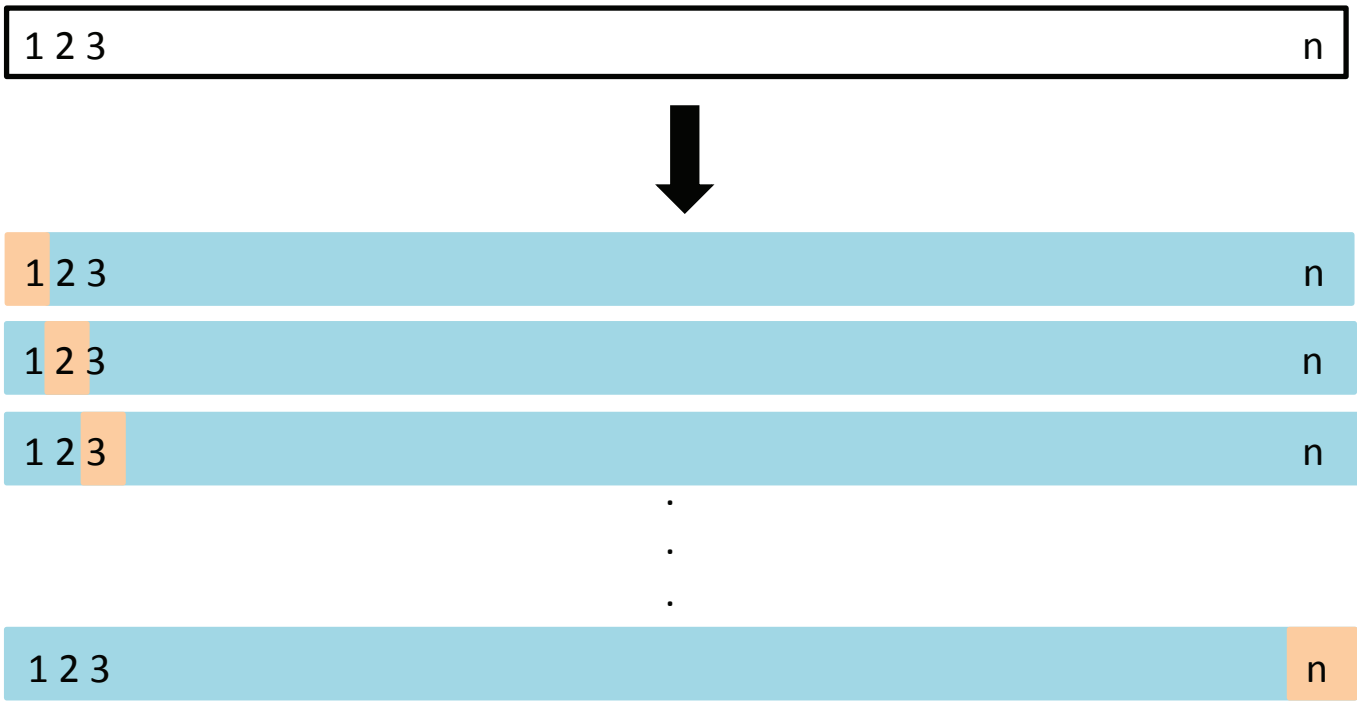

Figure: LOOCV approach

Note: A set of n observations is repeatedly split into a **training set** (in blue) containing all but one observation, and a **validation set** (in beige) that contains the remaining observation—the statistical learning method is fit on all but one observation i, $i = 1, \cdots, n$.

The LOOCV estimate for the test MSE is the average of the n test error estimates:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \widehat{\texttt{MSE}}_i$$

where $\widehat{\texttt{MSE}}_i$ is an estimate of the MSE when the training set contains all but observation i and the validation set only contains observation i:

$$\widehat{\texttt{MSE}}_i = (y_i - \widehat{y}_i)^2 .$$

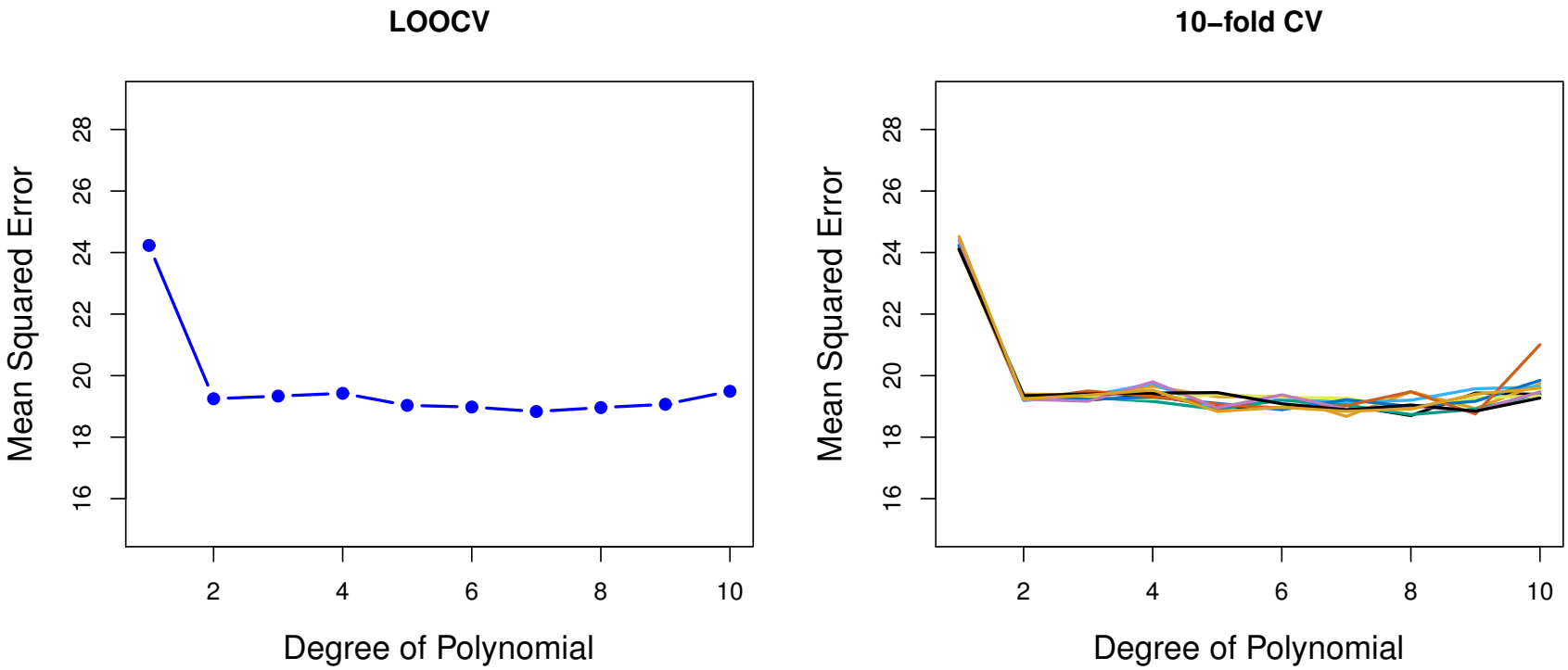

Figure: Implementation of a validation set approach

Note: Left panels correspond to the estimate of the LOOCV error curve. Right panels corresponds to a 10-fold CV.

Main insights

- LOOCV approach has several advantages w.r.t. validation set approach
  - ✓ Sample size: Make use of $n-1$ observations rather than (generally) half size of the original data set
  - ✓ Reduce the risk of overestimating the test error rate.
  - ✓ Absence of randomness in the training/validation set splits
- LOOCV approach can be expensive to implement especially for large n and complex statistical models!
- In the case of least squares or polynomial regression, one can show that

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{y}_i}{1-h_i}\right)^2$$

where $\widehat{y}_i$ is the i-th fitted value from the original least squares fit and h_i denotes the amount that an observation influences its own fit.

k-Fold Cross-Validation

- Step 1: Divide randomly the set of observations into k groups or folds (of approximately equal size);
- Step 2: The first fold is treated as a validation set and the statistical method is fit on the remaining $k-1$ folds;
- Step 3: Compute the (empirical) MSE for the first fold, $\widehat{\texttt{MSE}}_1$;
- Step 4: Repeat Step 2 and Step 3 for other folds.

The k-fold CV estimate is computed by averaging $\widehat{\texttt{MSE}}_i$'s ($i = 1, \cdots, k$):

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \widehat{\texttt{MSE}}_i$$

Remarks:

- In practise, one typically performs k-fold CV using $k = 5$ or $k = 10$ (as a rule-of-thumb).
- The main obvious advantage of k-fold CV w.r.t. LOOCV ($k = n$) is computational.
- In general, k-fold CV often gives more accurate estimates of the test error rate than LOOCV: this is related to the bias-variance trade-off:
  - LOOCV: "Small" bias (since estimation with $n-1$ observations) but "large" variance (average of n-fitted models on almost the same training sample: outputs are highly (positively) correlated)
  - k-fold CV: "Intermediate" bias but "small" variance (average of outputs of k-fitted models that are less correlated).
- k-fold CV leads to a variability that is much lower than the one in the test error estimates in the validation test approach.

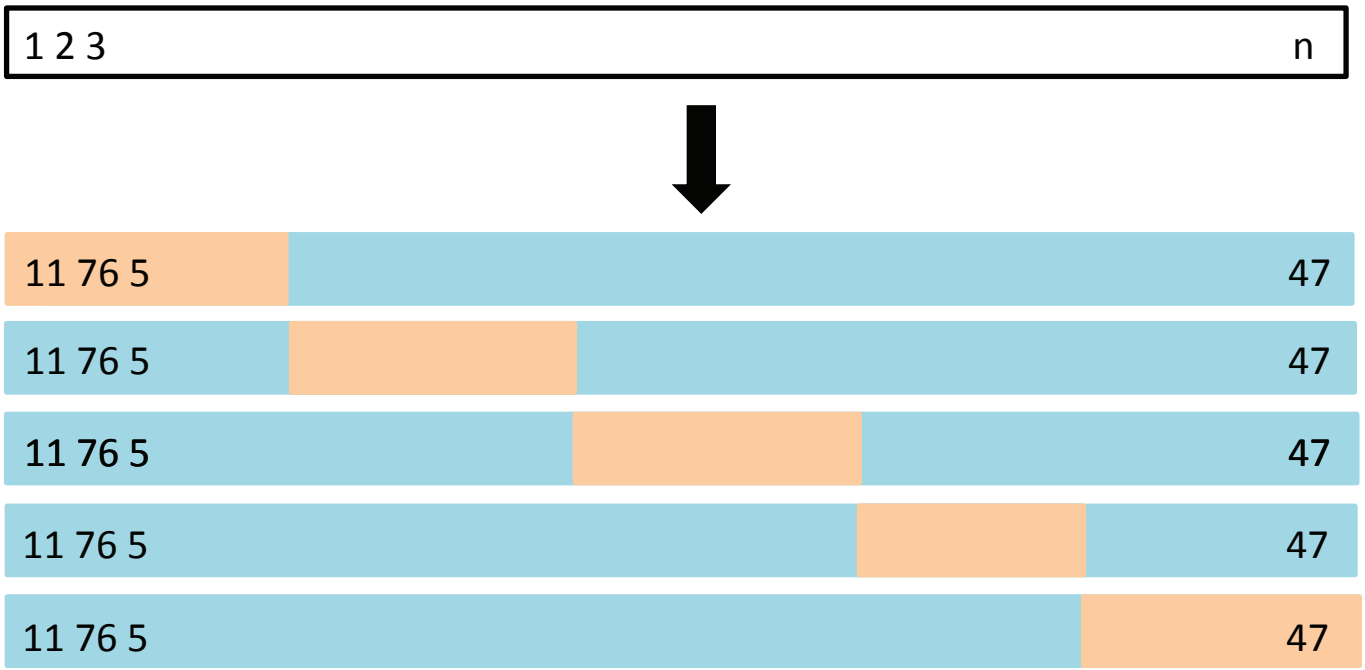

Figure: k-Fold CV approach

Note: A set of n observations is randomly split into five non-overlapping groups: each of this group subsequently acts as a **validation set** (in beige) and the remainders as a **training set**.

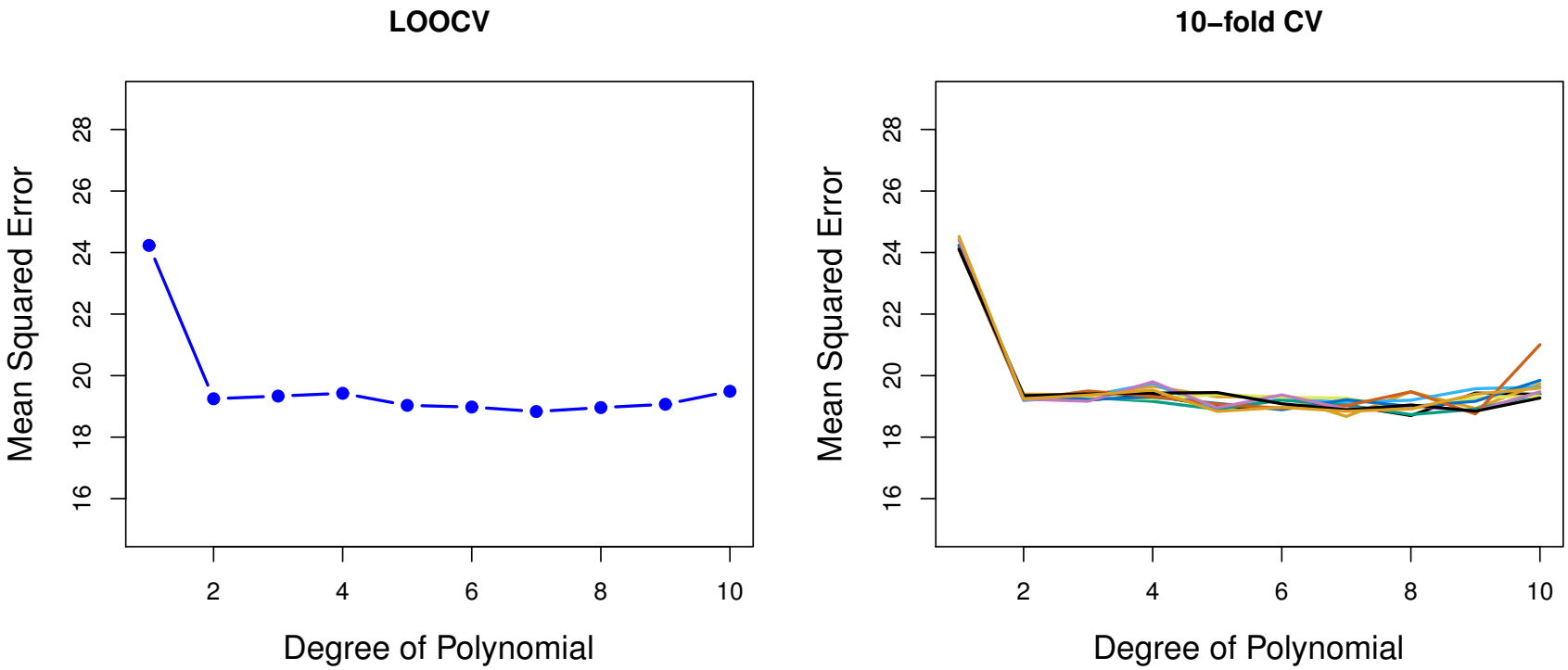

Figure: Implementation of a validation set approach

Note: Left panels correspond to the estimate of the LOOCV error curve. Right panels corresponds to a 10-fold CV.

References

- **(Working) Papers**

  - ✓ Boucheron, S., Bousquet, O. and G. Lugosi, 2005, Theory of Classification: A survey of some recent advances, ESAIM: Probability and Statistics, vol. 5, pp. 323-375;
  - ✓ Chakraborty; C., and A. Joseph, 2017, Machine Learning at Central Banks, Staff Working Paper NO. 647, Bank of England;
  - ✓ Kolanovic, M. and R.T. Krisnamachari, 2017, Big Data and AI Strategies, J.P. Morgan publication;
  - ✓ Mullainathan, S., and J. Spiess, 2017, Machine Learning: An Applied Econometric Approach, Journal of Economic Perspectives, vol. 31(2), pp. 87-106
  - ✓ Varian, H.R., 2014, Big Data: New Tricks for Econometrics, Journal of Economic Perspectives, vol. 28(2), pp. 3-28.

- **Textbooks**:
  - ✓ Hastie, T., Tibshirani, R. and J. Friedman, 2009, The Elements of Statistical Learning, 2nd edition, Springer Verlag.
  - ✓ Biau, G., and L. Devroye, 2015, Lectures on the Nearest Neighbor Method, Springer series in the Data Sciences, Springer Verlag.
  - ✓ Shawe-Taylor, J., and N. Cristianini, 2004, Kernel Methods for Pattern Analysis, Cambridge University Press.
  - ✓ Schölkopf, B., and J. Smola, 1998, Learning with Kernels, MIT Press.
  - ✓ Bühlmann, P. and S. Van de Geer, 2011, Statistics for High-Dimensional Data, Springer Series in Statistics, Springer Verlag.