Model Assessment and Selection

Axel Benner
Biostatistics, German Cancer Research Center
INF 280, D-69120 Heidelberg
benner@dkfz.de

Topics

- Estimation and Statistical Testing
  - Simulation
  - Bootstrap
  - Jackknife
  - Permutation

- Prediction
  - Jackknife
  - Cross-validation
  - Bootstrap

Model Assessment and Validation

- Data splitting and extensions
  - one time splitting (training set + assessment set)
  - (leave-one-out) cross-validation
  - K-fold cross-validation

- Resampling plans
  - Jackknife
  - Bootstrap

Model: a current approximation to complex relationships

Predictive Accuracy

Some models are used only for hypothesis testing

If used for prediction, need to consider accuracy of predictions

Calibration: observed responses agree with predicted responses

Discrimination: model is able, through the use of predicted responses, to separate subjects with low observed responses from those with high responses

$\longrightarrow$ Model assessment/validation to ascertain wether predicted values from the model are likely to accurately predict responses on future subjects or subjects not used to develop the model

- major failure: overfitting
- two modes of validation: internal vs external
  - external:
    (i) use a different set of subjects
    (ii) use first m for model training and $n - m$ for validation.
    problem: holding back data from model fitting results in lower precision and power !!
  - internal:
    (i) apparent (evaluate fit on same data used to create fit)
    (ii) data splitting and its extensions
    (iii) resampling methods

Overfitting

- Fitting a regression model with 20 patients and 20 variables (counting the intercept) will result in $R^2 = 1$ no matter what the variables are.
  A p-variable fit to $p + 1$ observations will perfectly predict Y (as long as no two observations have the same Y).

  Such a model will yield predictions that appear almost random with respect to responses on a different data set.

- Analyzing too many variables for the available sample size will not cause a problem with apparent predictive accuracy.

  But calibration or discrimination accuracy assessed on a new sample will suffer caused by multiple comparison problems and trying to estimate too many parameters from the sample.

Validation Methods

Need to use some validation method to honestly assess the likely performance of a model on a new series of subjects.

Validation Methods

- Data-Splitting: split sample into two parts at random:
  - Use first part to develop model
  - Use second part to measure predictive accuracy

  Is an honest method but assessment can vary greatly when take different splits
- Cross-validation: e.g., leave out 1/10 of subjects, develop model in 9/10, evaluate in 1/10, repeat 10 times and average
  Still not very precise way to measure accuracy
- Resampling plans
  - Jackknifing
  - Bootstrapping

    Bootstrap methods are more precise and do not require holding back data

Comments on Validation

Drawing samples from quantities such as residuals from the model to obtain a distribution that is conditional on input X.

This approach requires that the model be specified **correctly**, whereas the unconditional bootstrap does not.

The unconditional estimates are similar to conditional estimators except for very skewed or very small samples.

(Aggregate) Prediction Error

Example: Multiple linear regression

- $y_i = x_i^T\beta + \varepsilon_i$
  $\rightarrow$ least-squares prediction rule for a new covariate vector x_+: $\hat{y}_+ = x_+^T\hat{\beta}$
- How accurate is this prediction rule?
- Measure accuracy of prediction by squared error loss

$$(y_+ - \hat{y}_+)^2$$

- (aggregate) prediction error

$$D = \frac{1}{n}\sum_{i=1}^{n} E(Y_{+i} - x_i^T\hat{\beta})^2$$

  with $Y_{+i} = x_i^T\beta + \varepsilon_{+i}$
- Since D is unknown, an estimate of $\Delta = E(D)$ is needed

- Since

$$\begin{aligned} D &= \frac{1}{n}\sum var(Y_{+i}) + \frac{1}{n}\sum (x_i^T\beta - x_i^T\hat{\beta})^2 \\ &= \sigma^2 + \frac{1}{n}(\hat{\beta}-\beta)^T X^T X(\hat{\beta}-\beta) \end{aligned}$$

we get

$$\boxed{\Delta = E(D) = \sigma^2(1 + \frac{p+1}{n})}$$

Practically we have

$$\hat{\Delta} = s^2(1 + \frac{p+1}{n})$$

Comments

- This example is very special, since
  we need a correct model and constant variance

- It is not extendable to other prediction rules, e.g. classification

  $\longrightarrow$ **Cross-validation** or resampling methods can help

General notation

- Measure prediction error by loss function: $c(y_+, \hat{y}_+)$
- Prediction rule: $\hat{y}_+ = \mu(x_+, \hat{F})$
  where $\hat{F}$ is the empirical distribution function sampled from distribution F
- (Aggregate) prediction error

$$D = D(F, \hat{F}) = E[c(Y_+, \mu(X_+, \hat{F}))|\hat{F}]$$

$$\longrightarrow \text{Use } \Delta = \Delta(F) = E(D(F, \hat{F}))$$

- Apparent error (resubstitution error):

  Use the same data for prediction which was used for fitting the model

$$\Delta_{app} = D(\hat{F}, \hat{F}) = \frac{1}{n} \sum_{i=1}^{n} c(y_i, \mu(x_i, \hat{F}))$$

  Δ_{app} *underestimates the true* Δ *("it is downwardly biased")*

  For linear regression

$$\hat{\Delta}_{app} = \frac{1}{n} RSS$$

  and so we have

$$\boxed{E(\hat{\Delta}_{app}) = \sigma^2(1 - \frac{p+1}{n})}$$

  If $p = n - 1$ then $E(\hat{\Delta}_{app}) = 0$

- Data Splitting

  Separate data used to form the rule and data used to assess the rule:

  – training set $\{(x_i, y_i) : i \in S_t\}$
  – assessment set (test set) $\{(x_i, y_i) : i \in S_a\}$,

  represented by $\hat{F}_t$ and $\hat{F}_a$

$$\Delta_{ds} = D(\hat{F}_a, \hat{F}_t) = \frac{1}{n_a} \sum_{i \in S_a} c(y_i, \mu(x_i, \hat{F}_t))$$

- Leave-one-out Cross-Validation

  Training sets of size $(n-1)$ are taken and prediction rule is tested for a single observation:

$$\Delta_{cv} = \frac{1}{n}\sum_{i=1}^{n} c(y_i, \mu(x_i, \hat{F}_{-i}))$$

  where $\hat{F}_{-i}$ represents the data excluding the i-th case.

  Note the small bias of leave-one-out cv:
  "It differs from Δ by terms of order n^{-2} (whereas the apparent error differs by terms of order n^{-1})".

- K-fold Cross-Validation

  Small perturbations in the fitted model leaving out single observations can make $\hat{\Delta}_{cv}$ too variable

  $\longrightarrow$ leave out groups of observations; especially K disjoined groups

$$\Delta_{cv,K} = \frac{1}{n}\sum_{i=1}^{n} c(y_i, \mu(x_i, \hat{F}_{-k(i)}))$$

  where $\hat{F}_{-k(i)}$ represents the data excluding the group containing the i-th case.

- K-fold Cross-Validation (cont)
  - Good strategy: Take $K = min(\sqrt{n}, 10)$
    A size of at least $\sqrt{n}$ should perturb the data sufficiently to give small variance.
  - Problem: increasing bias (especially if K is small)!

    $\rightarrow$ Reduce bias by adjustment:

    Denote by $\hat{F}_{-k}$ the data with the k-th group omitted, $k = 1, ..., K$, and let p_k denote the proportion of the k-th group in the data set.

    $\rightarrow$

$$\Delta_{acv,K} = \Delta_{cv,K} \; + \; D(\hat{F}, \hat{F}) - \sum_{k=1}^{K} p_k D(\hat{F}, \hat{F}_{-k})$$

K-fold adjusted cross-validation

1. Fit the regression model to all cases, calculate predictions $\hat{y}_i$ from that model, and average the values of $c(y_i, \hat{y}_i)$ to get D.
2. Choose group sizes $m_1, ..., m_K$ such that $m_1 + ... + m_K = n$.
3. For $k = 1, ..., K$
   (a) choose C_k by sampling m_k times without replacement from {1,2,...,n} minus elements chosen for previous C_is
   (b) fit the regression model to all data except cases $i \in C_k$
   (c) calculate new predictions $\hat{y}_i = \mu(x_i, \hat{F}_{-k})$ for $i \in C_k$
   (d) calculate predictions $\hat{y}_{ki} = \mu(x_i, \hat{F}_{-k})$ for all i; then
   (e) average the n values $c(y_i, \hat{y}_{ki})$ to give $D(\hat{F}, \hat{F}_{-k})$.
4. Average the n values of $c(y_i, \hat{y}_i)$ using $\hat{y}_i$ from step 3(c) to give $\hat{\Delta}_{cv,K}$.
5. Calculate $\Delta_{acv,K} = \Delta_{cv,K} + D(\hat{F}, \hat{F}) - \sum_{k=1}^{K} p_k D(\hat{F}, \hat{F}_{-k})$ with $p_k = m_k/n$.

Misclassification error (two groups)

- Suppose a response y which is equal 1 or 0.
- The prediction rule $\mu(x_+, \hat{F})$ is an estimate of $P(Y_+ = 1|x_+)$ for a new case (x_+, y_+).
- Set $\hat{y}_+ = 1$ if $\mu(x_+, \hat{F}) \geq 0.5)$ and $\hat{y}_+ = 0$ otherwise.
- If misclassifications costs are equal the misclassification loss function is

$$c(y_+, \hat{y}_+) \;=\; \begin{cases} 1, & y_+ \neq \hat{y}_+ \\ 0, & \text{otherwise} \end{cases}$$

- The aggregate prediction error D is then the overall misclassification rate, equal to the proportion of cases where y_+ is wrongly predicted.

K-fold adjusted cross-validation

Example: The expression set Huang.RE which is discussed in THE LANCET (2003) 361:1590-1596. The data contains microarrays of 52 women with breast cancer of whom 34 did not experience a recurrence of the tumour during a 3 years time period.

Using probe set 34361_at in a logistic regression model with misclassification error loss we get

```
> print(delta.app) # apparent error
[1] 0.1923077
> print(delta.cv)  # leave-one-out cv
[1] 0.2115385
> print(delta.cv.k) # 7-fold cv
[1] 0.1923077
> print(delta.acv.k) # adjusted 7-fold cv
[1] 0.2067308
```

Drawbacks of cross-validation

- Choice of the number of observations to be hold out from each fit
- Number of repetitions needed to achieve accurate estimates of accuracy exceeds 200
  e.g. omit 1/10 of the sample 200 times to accurately estimate index of interest [sample need to be split into tens 20 times].
- Monte-Carlo cv is an improvement over ordinary cv (Picard & Cook, JASA, 1984)
- cv not fully represent variability of variable selection. if 20 subjects are omitted each time from set of 1000, list of variables selected from each sample of size 980 are likely to be different from lists obtained from independent samples of 1000 subjects. $\longrightarrow$ cv does not validate the full 1000 subject model.

Estimate Prediction Error (Bootstrap)

The bootstrap estimate of the prediction error is

$$\hat{\Delta} = \Delta(\hat{F}) = E(D(\hat{F}, \hat{F}^{*b}))$$

where $\hat{F}^{*b}$ denotes a bootstrap sample $(x_1^{*b}, y_1^{*b}), \ldots, (x_n^{*b}, y_n^{*b})$ of the original data.

Now the prediction rule is fitted to these data resulting in predictions $\mu(x_i, \hat{F}^{*b})$ of y_i.

Using a loss function $c(\cdot)$ $\hat{\Delta}$ is then approximated by

$$\hat{\Delta}_b \;=\; \frac{1}{B}\sum_{b=1}^{B}\frac{1}{n}\sum_{i=1}^{n} c(y_i, \mu(x_i, \hat{F}^{*b}))$$

derived by fitting the model on a set of bootstrap samples, and comparing its predictions with the original data.

Problem: Bootstrap sample act as training sample, and original training set act as test set.

Both samples have observations in common

$\longrightarrow$ overoptimistic estimate due to overfitting

$\longrightarrow$ underestimates the error

- Alternative 1: Leave-one-out bootstrap estimate of prediction error

$$\hat{\Delta}_{bcv} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|B_{-i}|}\sum_{b\in B_{-i}} c(y_i, \mu(x_i, \hat{F}^{*b}))$$

B_{-i} is set of indices that does not contain observation i and $|B_{-i}|$ is the size of this set.

Note that $|B_{-i}|/B$ is approximately equal to $e^{-1} = 0.368$

$\hat{\Delta}_{bcv}$ is a bootstrap smoothing of the leave-one-out cv.

$\longrightarrow$ overfitting no problem, but (like cv) bias by training set size.

$\longrightarrow$ possibly overestimates error rate.

Example: 6 bootstrap samples

original data	1	2	3	4	5
bootstrap sample 1	**1**	**1**	**3**	**4**	**4**
bootstrap sample 2	1	2	2	3	5
bootstrap sample 3	**1**	**3**	**3**	**3**	**4**
bootstrap sample 4	**3**	**4**	**4**	**5**	**5**
bootstrap sample 5	2	2	3	4	4
bootstrap sample 6	1	1	2	4	5

Now bootstrap samples 1,3,and 4 do not include observation 2.
And so we get: $B_{-2} = \{1, 3, 4\}$ with $|B_{-2}| = 3$.

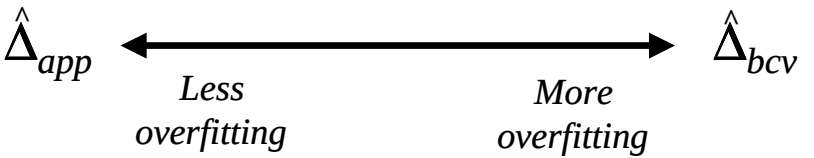
- Alternative 2: ".632" bootstrap estimate of prediction error

$$\hat{\Delta}_{.632} \;=\; .368\hat{\Delta}_{app} \;+\; .632\hat{\Delta}_{bcv}$$

where $\hat{\Delta}_{app}$ is the apparent error estimate

Pulls leave-one-out down toward training error

$\rightarrow$ may underestimate the error in overfitting situations



- Problem: $\hat{\Delta}_{.632}$ can break down in overfitted situation
  $\rightarrow$ take into account amount of overfitting.

  Let γ denote the non-information error rate

$$\hat{\gamma} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{i'=1}^{n} c(y_i, \mu(x'_i, \hat{F}^{*b}))$$

  i.e. the error rate if input and output are independent

  $\rightarrow$ relative overfitting rate

$$\hat{R} \;=\; \frac{\hat{\Delta}_{bcv} - \hat{\Delta}_{app}}{\hat{\gamma} - \hat{\Delta}_{app}}$$

  with $\hat{R} = 0$ if no overfitting occurs.

- And so:

  Alternative 3: ".632+" bootstrap estimate of prediction error

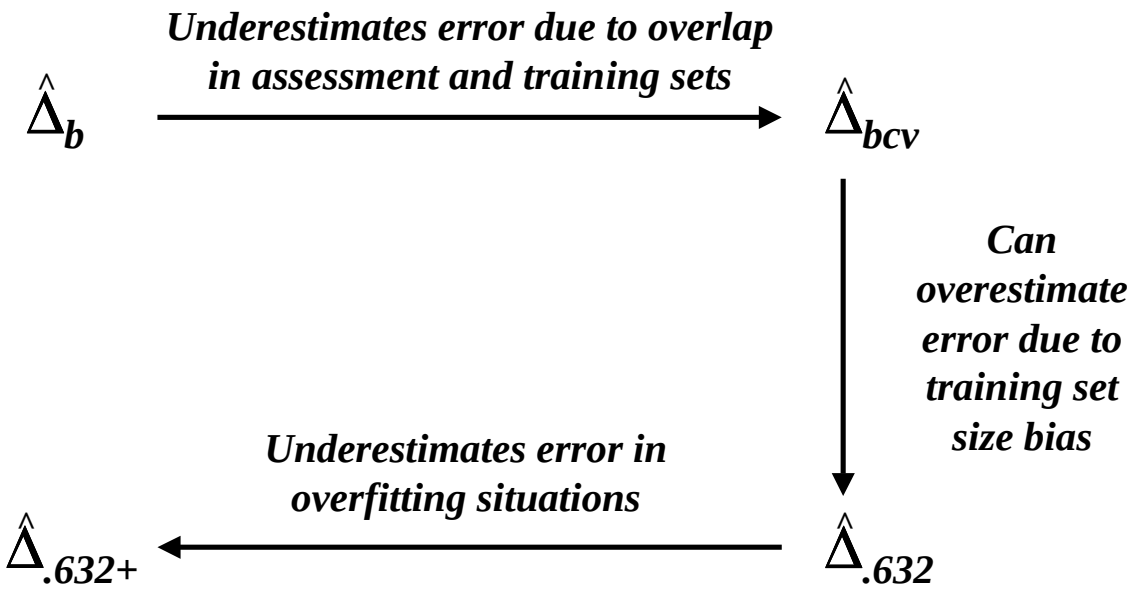
$$\hat{\Delta}_{.632+} \;=\; (1-\hat{w})\cdot\hat{\Delta}_{app} \;+\; \hat{w}\cdot\hat{\Delta}_{bcv}$$

  where

$$\hat{w} \;=\; \frac{.632}{1-.368\hat{R}}\;.$$

  $\hat{\Delta}_{.632+}$ varies from $\hat{\Delta}_{.632}$ to $\hat{\Delta}_{bcv}$.

Bootstrap overview



Misclassification error (two groups)

- The prediction $\mu(x_+, \hat{F})$ and the measure of error $c(y_+, \hat{y}_+)$ are not continuous functions of the data.

  $\rightarrow$ bootstrap methods for estimating D or its expected value Δ are superior to cross-validation methods, in terms of variability.

Variable/Gene Selection

- Model/variable selection implies that there is some likelihood of a "true" model,

  some pre-specified variables have zero association with response Y

- Need to perform gene selection preceding the predictive modelling

  $\longrightarrow$ e.g. eliminate variables whose distributions are too narrow.

Variable/Gene Selection (cont)

- Gene filtering is helpful, but

  estimating the error rate after variable selection leads to biased estimates of the prediction error
  $\rightarrow$ overstating importance of variables which are retained in the model.

- Make sure that you are cross-validating the experiment that you have carried out,
  in particular, if you are selecting genes, rather than working with known genes, you must cross-validate the gene selection process as well.

- There are many examples with low classification error rates which do not cross-validate properly (model/gene selection was not validated).

Prediction error in gene selection situations

For simplicity select 1000 most variable probe sets (e.g. by largest variability) for the exercises (data frame `mydata`)

```
library(affy)
sd.exp <- apply(exprs(Huang.RE),1,sd)
index <- order(sd.exp,decreasing=TRUE)[1:1000]

mydata <- data.frame(t(exprs(Huang.RE)[index,]),
                     Recurrence=as.factor(pData(Huang.RE)$Recurrence))
```

Now we select probe sets by comparing their univariate p-values of a two-sample t-test with a pre-specified level of 0.05 and train a LDA using the selected probe sets only (function `mymod`).

```
mymod <- function(formula, data, level = 0.05) {
  sel <- which(lapply(data, function(x) {
     if (!is.numeric(x))
          return(1)
     else return(t.test(x ~ data$Recurrence)$p.value)
  }) < level)
  sel <- c(which(colnames(data) %in% "Recurrence"), sel)
  mod <- lda(formula, data = data[, sel])
  function(newdata) {
  predict(mod, newdata = newdata[, sel])$class
 }
}
```

The **.632+ bootstrap** estimate of the prediction error using B=25 bootstrap samples gives a misclassification error of 0.27.

```
library(ipred)
set.seed(71003)
errorest(Recurrence ~ ., data=mydata, model=mymod, estimator="632plus",
                         est.para=control.errorest(nboot=25))
```

```
errorest.data.frame(formula=Recurrence ~ ., data=mydata,
    model=mymod, estimator="632plus", est.para=control.errorest(nboot=25))

        .632+ Bootstrap estimator of misclassification error
        with 25 bootstrap replications

Misclassification error:  0.2705
```

Define a gene expression set of 1000 genes with no association to the response

```
set.seed(63321)
mydata <- data.frame(matrix(rnorm(52*1000),52,1000),
                     Recurrence=as.factor(pData(Huang.RE)$Recurrence))
```

1. Select genes by individual t tests (selection level 0.05), perform a **lda** using the selected subset and compute estimate of the misclassification error (ignoring the selection process)

```
sel <- which(lapply(mydata, function(x) {
     if (!is.numeric(x)) return(1)
     else return(t.test(x ~ mydata$Recurrence)$p.value)
}) < 0.05)
sel <- c(which(colnames(mydata) %in% "Recurrence"), sel)
errorest(Recurrence ~ ., data = mydata[, sel],
 model = lda, estimator = "632plus", predict = mypredict.lda)
```

```
Call:
errorest.data.frame(formula = Recurrence ~ ., data = mydata[,
    sel], model = lda, predict = mypredict.lda, estimator = "632plus")

        .632+ Bootstrap estimator of misclassification error
        with 25 bootstrap replications

Misclassification error:  0.1005
```

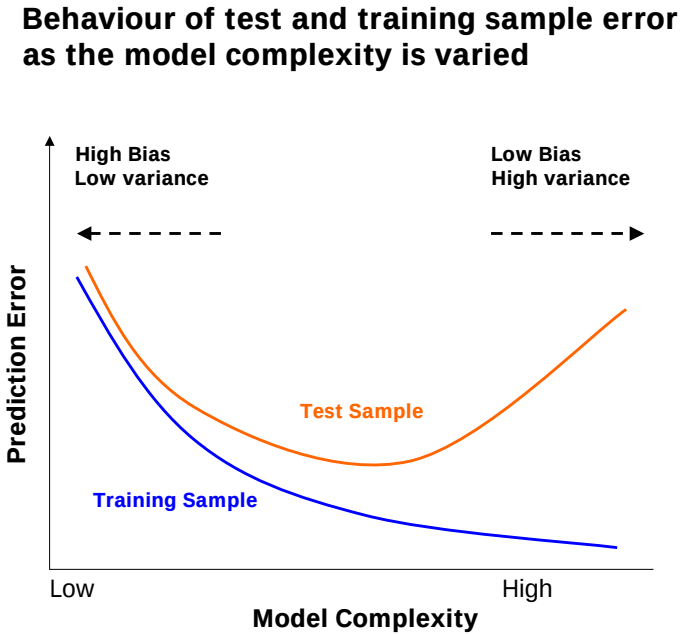
2. Now repeat the error estimation taking into account the gene selection by individual t tests (using 25 bootstrap samples)

```
errorest(Recurrence ~ ., data=mydata, model=mymod,
    estimator="632plus", est.para=control.errorest(nboot=25))

Call:
errorest.data.frame(formula = Recurrence ~ ., data = mydata,
    model = mymod, estimator = "632plus",
    est.para = control.errorest(nboot = 25))

        .632+ Bootstrap estimator of misclassification error
        with 25 bootstrap replications

Misclassification error:  0.3447
```

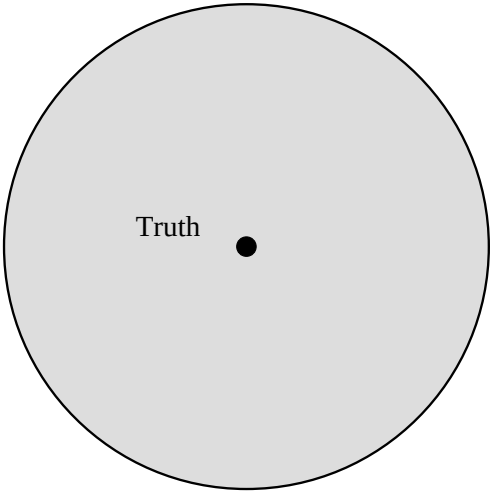
Result of this example:

Ignoring the selection process results in an error estimate of 10% which is about 1/3 of the resulting error estimate if we include the pre-selection in the error estimation procedure.

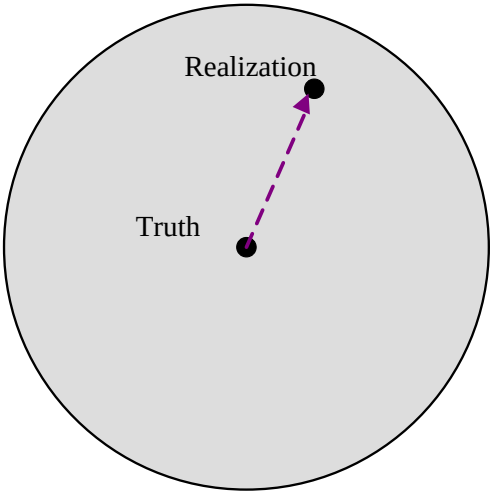
Problem of overfitting



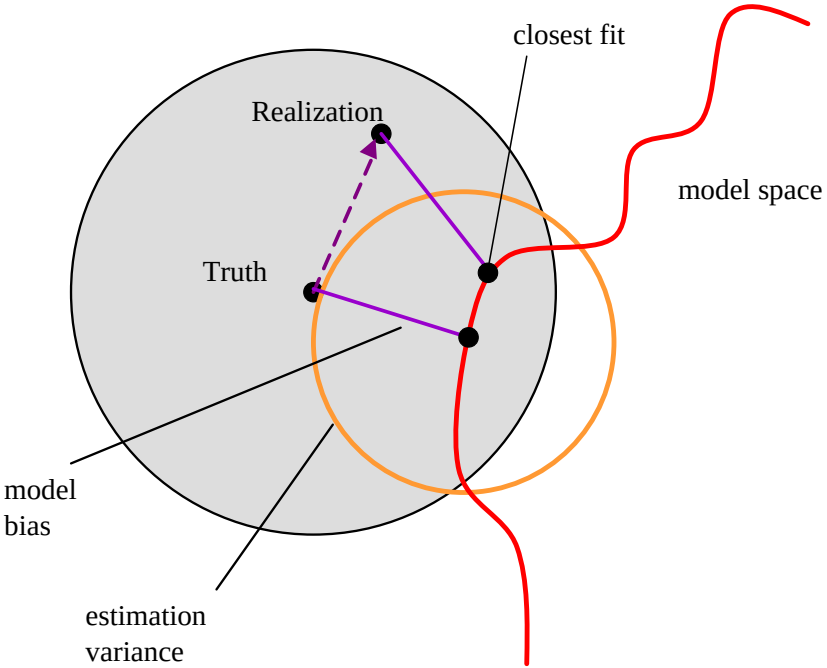
Controlling the complexity of the model



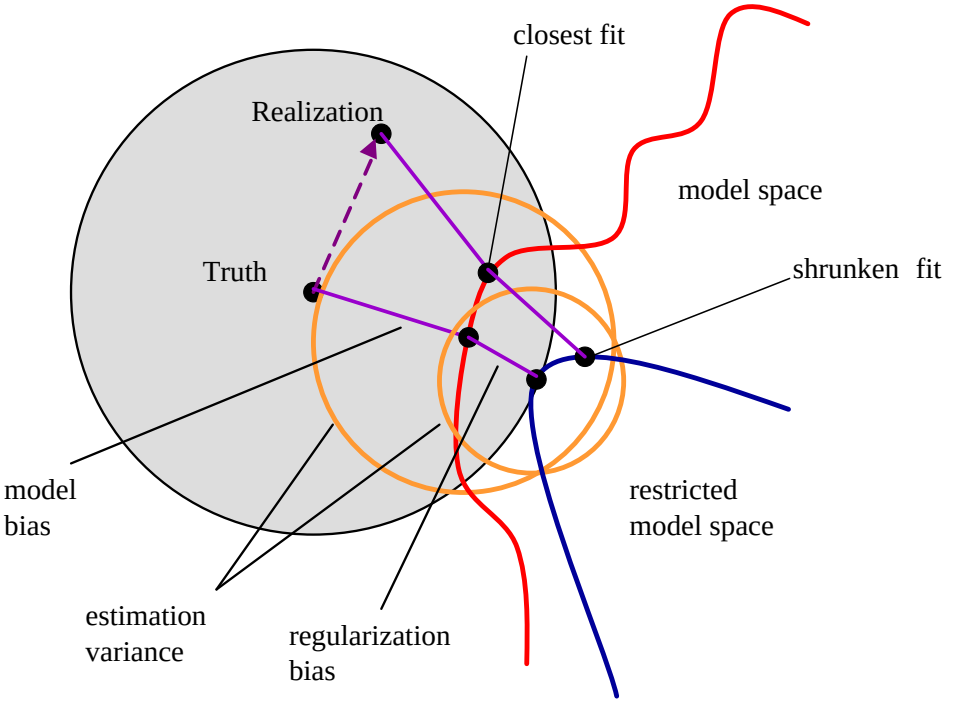
Hastie, Tibshirani, Friedman, 2001



Hastie, Tibshirani, Friedman, 2001



Hastie, Tibshirani, Friedman, 2001



Hastie, Tibshirani, Friedman, 2001

Controlling the complexity of the model

- **Restriction methods**

  The class of functions of the input variables defining the model is limited.

  Example:
  Allow only linear combinations of given basis functions h_{jm}

$$f(X) = \sum_{j=1}^{G} f_j(X_j) = \sum_{j=1}^{G} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j)$$

  h_{jm} is the m^{th} basis function of the j^{th} input variable.

  The size of the model is limited by the number M_j of basis functions used for the G components f_j.

Controlling the complexity of the model

- **Selection methods**

  Include only those basis functions h_{jm} that contribute 'significantly' to the fit of the model.

  Examples:

  – Variable selection methods

  – Stagewise greedy approaches like boosting

- **Regularization methods**

  Restrict the coefficients of the model.

  Example: Ridge regression

Penalized Regression

- Maximizing the log likelihood can result in fitting noise in the data.
- A shrinkage approach will often result in estimates of the regression coefficients that, while biased, are lower in mean squared error and are more close to the true parameters.
- A good approach to shrinkage is penalized maximum likelihood estimation (le Cessie & van Houwelingen, 1990).

  From the log-likelihood $logL$ a so-called 'penalty' is subtracted, that discourages regression coefficients to become large.

  $\rightarrow$ penalized log likelihood:

$$logL - \lambda \cdot p(\beta)$$

  $p(\beta)$ penalty function, λ non-negative penalty factor.