DIPARTIMENTO
DI STATISTICA PROBABILITÀ
E STATISTICHE APPLICATE

SAPIENZA
UNIVERSITÀ DI ROMA

Technical Report 2008/19.

Estimators of extra-sample error for non-parametric methods. A comparison based on extensive simulations.

Simone Borra

Università di Roma "Tor Vergata", borra@uniroma2.it

Agostino Di Ciaccio

Università di Roma "La Sapienza", agostino.diciaccio@uniroma1.it

Available on-line from 27/12/2008
http://www.dspsa.uniroma1.it/on-line/Home/Ricerca/Pubblicazioni.html

Abstract
In this technical report we examine and compare the estimators most widely used to evaluate the prediction error of a non-parametric regression model. An extensive simulation approach allowed us to compare the estimators with respect to different data characteristics: the linearity of the relation, the signal-to-noise ratio, and the amount of model overfitting.
We compared estimators based on resampling methods such as Leave-one-out, parametric and non-parametric Bootstrap, as well as repeated Cross Validation methods and Hold-out. The models used were Regression Trees, Projection Pursuit Regression and Neural Networks. The simulations show that the repeated corrected 10-fold cross-validation proposed by Burman and the Parametric Bootstrap proposed by Efron have the best performance.

Keywords: Extra sample error, Optimism, Cross-Validation, Resampled Hold-Out, Bootstrap .632+, Leave-One-Out, Parametric Bootstrap, Prediction Error, Regression Trees, Projection Pursuit Regression, Neural Networks.

Table of contents

1. Introduction

Given a predictive statistical model, estimated on the available sample data, a fundamental problem in statistics is that of obtaining an accurate estimate of the prediction error, i.e. the expected loss of the model on future observations.
This task has particular relevance every time a very large sample is not available, the underlying distribution is not known and you need to evaluate the prediction error of a non-parametric model which could overfit data.
It is quite common to meet this condition in Data Mining problems, where objective of the analysis is often to predict correctly the values of a target variable on new observed cases. For this aim, we should select the model having the greatest prediction capability and this requires to evaluate each model, ensuring the evaluation comparability with other concurrent models, which could have a different parametrization. Consequently we need a reliable prediction-error estimator which does not assume a specific model and which does not require restrictive hypotheses on the data generation process.

In literature the main effort, regarding the model evaluation, refers to the choice of the best model in a fixed homogeneous class (f.e. the linear regression models). In that case we don't need a precise assessment of each model because it is sufficient to use a "relative measure" comparing the (usually nested) models to select, for example, the most important explicative variables. With this respect, the most famous proposals are the Mallows's Cp, Akaike's information criterion (AIC) and the Bayesian information criterion (BIC) (Mallows 1973, Akaike 1973, Schwarz 1978) or the Adaptive Model Selection based on a concept of generalized degrees of freedom proposed by Shen & Ye (2002).
Alternative approaches for model selection are based on cross-validation or bootstrap techniques producing many estimators of prediction error. In literature the most relevant proposals consider mainly linear or near-linear models (see Shao 1993, Zhang 1993, Efron 2004) and the asymptotic properties of the estimators (Stone 1977, Dudoit & van der Laan 2005). Shao (1997, 1993) showed that in linear models, leave-one-out estimator is asymptotically equivalent to AIC and Mallows's Cp but it is asymptotically inconsistent in the sense that the probability of selecting the model with the best predictive ability does not converge to 1 as the number of observations goes to infinity.

On the other hand, if we want to evaluate the predictive performance of a model or select the best model in a heterogeneous class of parametric and non-parametric models, we need a reliable assessment of a model prediction capability.
There are some results available providing bounds on the accuracy of the various estimators of prediction error (see f.e. Devroye, Györfi and Lugosi, 1996). Perhaps the most *general* results are those given for the (classification) training error estimate by Vapnik (1998), who proved that for any target function and input distribution, and for any learning algorithm that chooses its hypotheses from a class of VC dimension *d*, the training error estimate is at most $\tilde{O}\left(\sqrt{d/m}\right)$ away from the true error, where *m* is the size of the training sample. On the other hand, among the *strongest* distribution free performance bounds are those given by Devroye and Wagner (1979), that can be applied to linear discrimination rules, nearest neighbours and histogram rules, and by Gascuel and Caraux (1992). Others bounds for the leave-one-out estimate have been proposed, for example by Rogers and Wagner (1978) for local discrimination rules, by Zhang (2001) for kernel methods and by Kearns & Ron (1999). However all the results are limited to classification problems, and the bounds are not sharp with respect to realistic situations. As alternative to the classical complexity-oriented

approach, recent approaches to deriving generalization bounds are based on the algorithmic stability, that is the behaviour of a learning algorithm with respect to perturbations of the training set (see f.e. Bousquet and Elisseeff, 2002).
For the above reasons, a good generalization and a consistent model selection could lead to conflicting decision rules regarding the best choice of the prediction error estimator (see Larsen and Goutte, 1999).

Efron (1986) introduced the *optimism* theorem and, therefore, the problem of estimating the *prediction error* became a problem of estimating covariance-penalty terms. Recently, Efron (2004) proposed a new parametric bootstrap approach, moreover new developments in the covariance-penalty approach for nonparametric regression and classification have been provided by Zhang (2008) and Daudin & Mary-Huard (2008).

To the best of our knowledge, ours is the first large-scale study on the performance of *prediction error* estimators taking into consideration non-parametric regression models. More precisely, we propose carrying out the following tasks:

- compare the most widely known estimators of *prediction error* with respect to different nonparametric regression models;
- analyse the importance of sample-size, non-linearity, overfitting, and the signal-to-noise ratio to explain the estimators' performance on different data-sets.

We deal with these specific tasks in the next paragraphs using an extensive simulation approach without considering any restrictive assumptions.

In section 2 we introduce the concept of *extra-sample error*, *in-sample error*, expected *extra-sample error* and their relationships.
The concepts of *apparent error* estimator and *optimism* are introduced in section 3.
In section 4 we introduce the estimators based on cross-validation, while the estimators based on repeated cross-validation are introduced in section 5. In section 6 and 7 we introduce the estimators based on bootstrap, parametric and non parametric, including the improvements 632 and 632+. Section 8 considers a tentative of unified view of prediction error estimators proposed in literature. In section 9 we show the complex structure of each simulation and the evaluation measures employed to compare the estimators. The characteristics of each simulation carried out, are explained in section 10 with the results of the comparison of all the estimators. To reinforce the previous results, we considered also the analysis with an ill-specified model in section 11 where the usual hypotheses for the noise (homoschedasticity, normal distribution with zero mean) do not hold. Finally in the last paragraph we provide some conclusions.

2. The Prediction Error

Let $Y = f(X_1, X_2 ....., X_J) + \varepsilon$, where f is a general unknown function of the covariates $\mathbf{X} = X_1, X_2 ....., X_J$ and ε is a random noise with zero mean and variance σ^2.

Let

$$\mu = f(x_1, x_2 ....., x_J) = E(Y \mid \mathbf{x}) \tag{1}$$

be the conditional expected value of Y given the value of covariates. We can indicate the conditional distribution of Y for the i-th observation as $(Y_i \mid \mathbf{x}_i) \sim G_{\mu_i}$. If we suppose $\varepsilon \sim N(0; \sigma_\varepsilon^2)$ then

$$(Y_i \mid \mathbf{x}_i) \sim N(\mu_i; \sigma_\varepsilon^2) \tag{2}$$

Having chosen an appropriate loss function $L(.)$ and given a sample $\mathbf{s}$ we can estimate the function $f(\mathbf{X})$ using the entire sample or, more generally, a subset, the *training-set* $\mathbf{c} = \{y_i, \mathbf{x}_i\}_1^n \subseteq \mathbf{s}$, obtaining the $\hat{f}_\mathbf{c}$. We also indicate the training set as $\mathbf{c} = (\mathbf{y}_\mathbf{c}, \mathbf{X}_\mathbf{c})$, where $\mathbf{y}_\mathbf{c}$ is the vector of observed Y_i and $\mathbf{X}_\mathbf{c}$ is the matrix of the covariate values.

Given $\mathbf{c}$ and estimated the function $\hat{f}_\mathbf{c}$, we want to know the prediction capability of $\hat{f}_\mathbf{c}$ on all the possible values $\mathbf{x}$ of $\mathbf{X}$.

The *prediction error* of a fixed $\hat{f}_\mathbf{c}$ is given by

$$Err(\hat{f}_\mathbf{c}) = E_\mathbf{x} E_Y [L(Y, \hat{f}_\mathbf{c}(\mathbf{x})) \mid \hat{f}_\mathbf{c}] \tag{3}$$

which in the following we will indicate simply as Err.
The *expected prediction error* is obtained averaging with respect to the training sample $\mathbf{c}$:

$$\overline{Err} = E_\mathbf{c} \{Err\} \tag{4}$$

Note that $E_\mathbf{c}$ averages on training samples, non fixing $\hat{f}_\mathbf{c}$, while $E_\mathbf{x}$ and E_Y consider the possible values of $\mathbf{X}$ and Y with a fixed $\hat{f}_\mathbf{c}$. Err is also called *generalization error*, and $\overline{Err}$ the *average generalization error*.

If we want to obtain a measure of the capability of a function $\hat{f}_\mathbf{c}$ estimated on the observed training set $\mathbf{c}$, we are mainly interested in Err. On the other hand, expectation (4) could be useful in obtaining a general evaluation of the given model, independently from a particular sample.

Expression (3) is also called *extra-sample error* (Efron & Tibshirani, 1997) because the average $E_\mathbf{x}$ refers to all possible values of $\mathbf{X}$.

Define the *expected prediction error* at the generic point $\mathbf{x}_o$ as

$$\overline{Err}(\mathbf{x}_o) = E_\mathbf{c} \{E_{Y_o} [L(Y_o, \hat{f}_\mathbf{c}(\mathbf{x}_o)) \mid \hat{f}_\mathbf{c}]\} \tag{5}$$

If we choose the quadratic loss function $L(Y_o, \hat{f}_\mathbf{c}(\mathbf{x}_o)) = (Y_o - \hat{f}_\mathbf{c}(\mathbf{x}_o))^2 = (Y - \hat{\mu}_{o\mathbf{c}})^2$

$\overline{Err}(\mathbf{x}_o) = E_\mathbf{c} E_{Y_o} (Y_o - \hat{\mu}_{o\mathbf{c}})^2 = E_\mathbf{c} E_{Y_o} (Y_o - E_\mathbf{c}(\hat{\mu}_{o\mathbf{c}}) + E_\mathbf{c}(\hat{\mu}_{o\mathbf{c}}) - \hat{\mu}_{o\mathbf{c}})^2 =$

$$= E_{\mathbf{c}} E_{Y_0} \left\{ (Y_0 - E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}))^2 + (E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}) - \hat{\mu}_{0\mathbf{c}})^2 + 2(Y_0 - E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}))(E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}) - \hat{\mu}_{0\mathbf{c}}) \right\} =$$
$$= E_{\mathbf{c}} E_{Y_0} (Y_0 - E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}))^2 + E_{\mathbf{c}} E_{Y_0} (E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}) - \hat{\mu}_{0\mathbf{c}})^2 + 0 =$$
$$= E_{Y_0} (Y_0 - E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}))^2 + E_{\mathbf{c}} (E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}) - \hat{\mu}_{0\mathbf{c}})^2 =$$
$$= E_{Y_0} (Y_0 - \mu_0 + \mu_0 - E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}))^2 + E_{\mathbf{c}} (E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}) - \hat{\mu}_{0\mathbf{c}})^2 =$$
$$= E_{Y_0} (Y_0 - \mu_0)^2 + (\mu_0 - E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}))^2 + E_{\mathbf{c}} (E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}) - \hat{\mu}_{0\mathbf{c}})^2 =$$
$$= \sigma_\varepsilon^2 + (\mu_0 - E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}))^2 + E_{\mathbf{c}} (E_{\mathbf{c}}(\hat{\mu}_{0\mathbf{c}}) - \hat{\mu}_{0\mathbf{c}})^2 =$$
$$= \sigma_\varepsilon^2 + \textit{predictor bias}^2 + \textit{predictor variance}$$

Then, the *Expected Prediction Error* at a generic input point $\mathbf{x}$ is

$$\overline{Err}(\mathbf{x}) = \sigma_\varepsilon^2 + \textit{predictor bias}^2 + \textit{predictor variance} \tag{6}$$

where σ_ε^2 is the variance of the noise, sometimes called the *irreducible error*.

A more restrictive definition of *prediction error* is the *in-sample error* of $\hat{f}_{\mathbf{c}}$, where the values of the covariates $\mathbf{x}_i \in \mathbf{X}_{\mathbf{c}}$, $\mathbf{y}_{\mathbf{c}}$ and $\hat{f}_{\mathbf{c}}$ are considered fixed at their observed sample values while Y_i are random variables as defined in (2), thus

$$Err_{in} = \frac{1}{n} \sum_{i=1}^{n} E_{Y_i} \left[L(Y_i, \hat{f}_{\mathbf{c}}(\mathbf{x}_i)) \,\middle|\, \hat{f}_{\mathbf{c}} \right] \tag{7}$$

Expression (7) has been considered by several authors mainly in a model selection approach (Efron, 1986, 2004; Ye, 1998; Shen & Ye, 2002).

On the other hand, considering the *in-sample error* approach, we can define the *expected in-sample prediction error* at the sample point $\mathbf{x}_i \in \mathbf{X}_{\mathbf{c}}$ as:

$$\overline{Err}_{in}(\mathbf{x}_i) = E_{Y_{\mathbf{c}}} \left\{ E_{Y_i} \left[L(Y_i, \hat{f}_{\mathbf{c}}(\mathbf{x}_i)) \,\middle|\, \hat{f}_{\mathbf{c}} \right] \right\} \tag{8}$$

Note that $\hat{f}_{\mathbf{c}}$ in (8) is not fixed: $E_{Y_{\mathbf{c}}}$ averages with respect to training samples which have fixed $\mathbf{X_c}$ (the values of the covariates in the training set) but different $\mathbf{y}_{\mathbf{c}}$.

The general expression of *expected in-sample prediction error* is obtained averaging with respect to the training units:

$$\overline{Err}_{in} = \frac{1}{n} \sum_{i=1}^{n} \left\{ \overline{Err}_{in}(\mathbf{x}_i) \right\} \tag{9}$$

If we choose the quadratic loss function, expression (8) can be written as

$$\overline{Err}_{in}(\mathbf{x}_i) = \sigma_\varepsilon^2 + (\textit{predictor bias}^2 \mid \mathbf{X}_{\mathbf{c}}) + (\textit{predictor variance} \mid \mathbf{X}_{\mathbf{c}}) \tag{10}$$

where the covariate values in the new training sets are fixed at their observed values $\mathbf{X_c}$.

We expect that by increasing the sample size, the difference between (3) and (7) will decrease because $\mathbf{X_c}$ includes more $\mathbf{x}_i$ of the generating distribution.

In the plots 2.1 and 2.2 we can see the *extra-sample error* (3) and the expected *in-sample error* (9), computed on 30.000 samples drawn from 1000 generating distributions. The predictive model is the Project Pursuit Regression (for details see section 9). It is evident that for a large sample (500 cases) the difference between the two measures is greatly reduced.
We can also see that the difference between *in-sample error* and expected *in-sample error* is quite modest also for a small sample size, as shown in the figure 2.3.

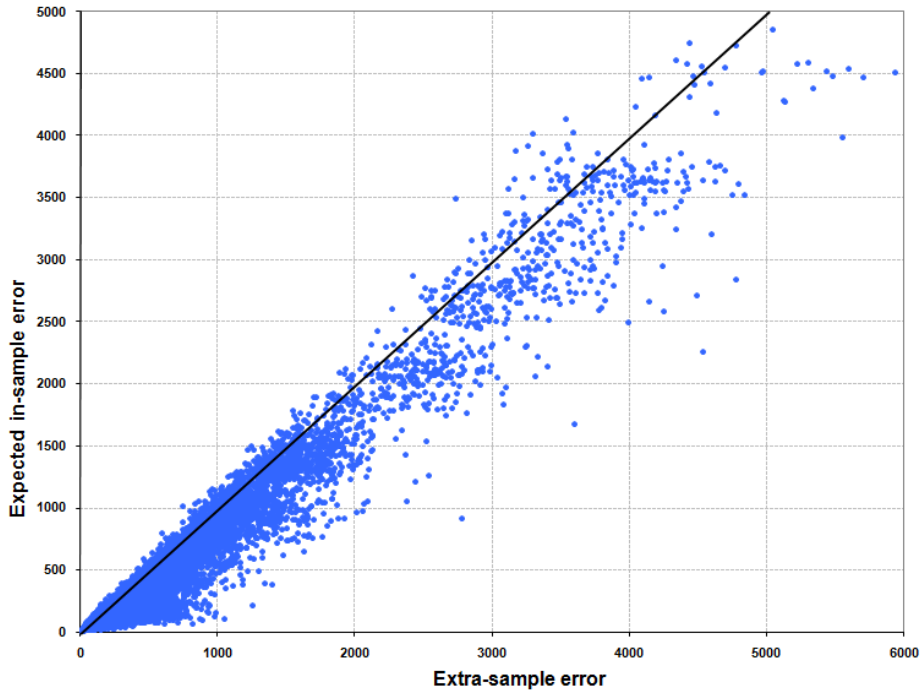

Fig. 2.1 – *Extra-sample error* vs expected *in-sample error*. PPR with n=120, number of samples=30000.

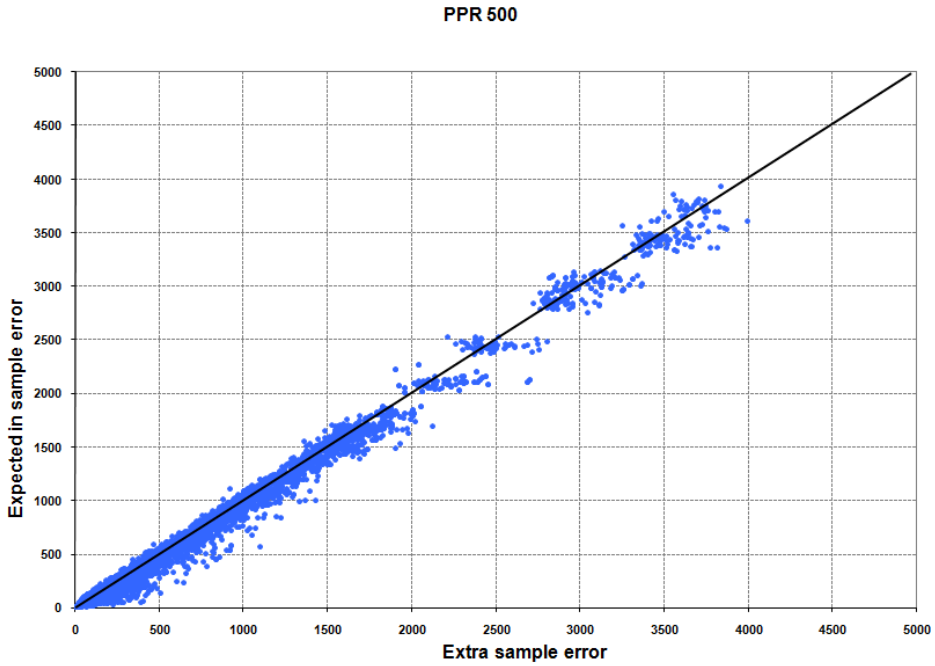

Fig. 2.2 – *Extra-sample error* vs expected *in-sample error*. PPR with sample size n=500, number of samples=30000.

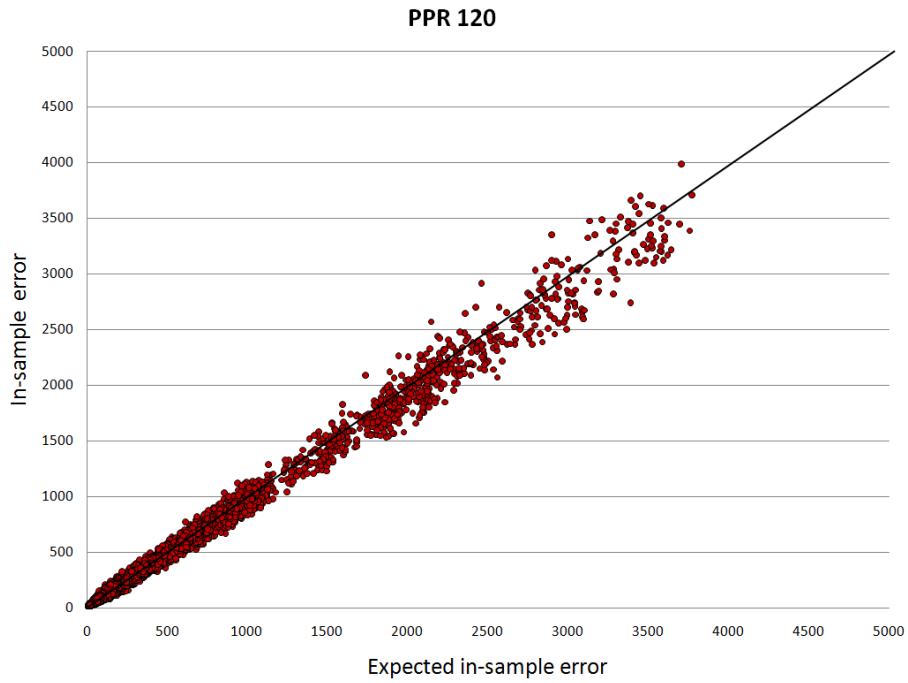

Fig. 2.3 – *In-sample error* vs expected *in-sample error*. PPR with n=120, number of samples=30000.

Also using a different model we can observe the same results, as shown in the fig. 2.4 and 2.5, where we applied regression tree models (for details see section 9).

If $f(\mathbf{X})$ is a linear function with p parameters, expression (10) can be written as (Hastie et al. 2001)

$$\overline{Err_{in}}(\mathbf{x}_i)=\sigma_\varepsilon^2+(\mu_i-E_{\mathbf{y}}(\hat{\mu}_{i\mathbf{c}}))^2+\|h(\mathbf{x}_i)\|^2\sigma_\varepsilon^2 \tag{11}$$

where $\mathbf{x}_i \in \mathbf{X}_\mathbf{c}$, $h(\mathbf{x}_i)$ is the vector of linear weights that produce the fit $\hat{\mu}_{i\mathbf{c}}=\hat{f}_\mathbf{c}(\mathbf{x}_i)=\mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}=h(\mathbf{x}_i)\mathbf{y}$, then $\text{var}(\hat{f}_\mathbf{c}(\mathbf{x}_i))=\|h(\mathbf{x}_i)\|^2\sigma_\varepsilon^2$. The average of this quantity with respect to the training data is $\frac{p}{n}\sigma_\varepsilon^2$. Finally we obtain:

$$\frac{1}{n}\sum_i \overline{Err_{in}}(\mathbf{x}_i)=\sigma_\varepsilon^2+\frac{1}{n}\sum_i(\mu_i-E_{Y_\mathbf{c}}(\hat{\mu}_{i\mathbf{c}}))^2+\frac{p}{n}\sigma_\varepsilon^2 \tag{12}$$

From expressions (6), (10) and (11) we can observe the bias-variance trade-off (Breiman 1992): minimizing the bias may increase the variance and vice-versa. In fact, in order to minimize the variance we could take a constant function $\hat{y}_i=c$, in this way the variance will be necessarily equal to zero while the bias will be very high. On the other hand, if our model interpolates perfectly the sample data (it is possible if there are not equal values of x_i) then we have:

$$E(\hat{Y}_i)=f(x_i) \quad \text{then} \quad B(\hat{Y}_i)^2=E\left[(f(x_i)-E(\hat{Y}_i))^2\right]=0$$

While $Var(\hat{Y}_i)$ will be equal to $Var(\varepsilon)$, which could be high.

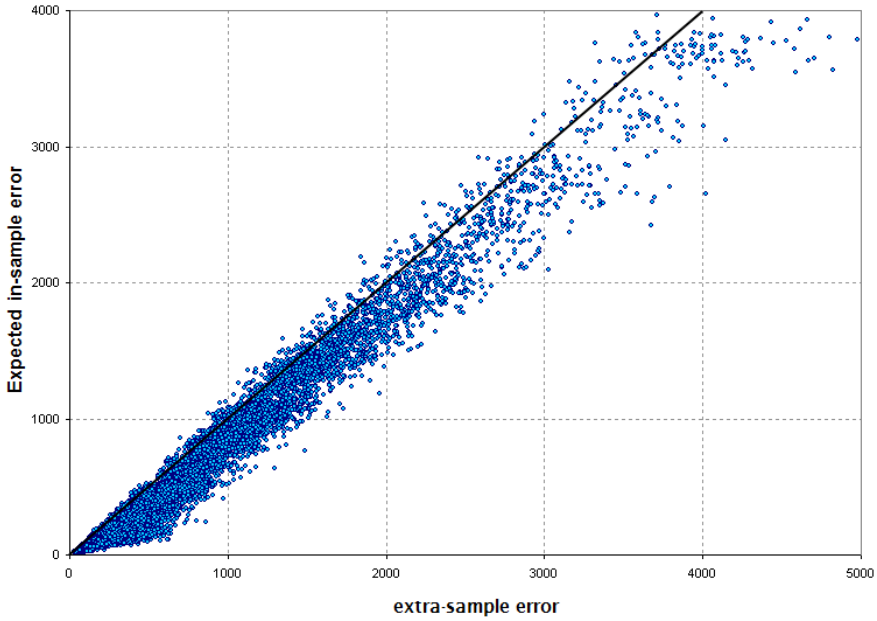

Fig. 2.4 – *Extra-sample error* vs expected *in-sample error*. TREES with n=120, number of samples=30000.

The use of overly complex models (overfitting) produces low distortion and high variability while the use of overly simply models (underfitting) may produce high distortion and low variability. Specifically, if the signal-to-error ratio is high, overfitting should not be a matter of concern because the *prediction error* will be at most double the error variance while underfitting could result in a much higher *prediction error*.

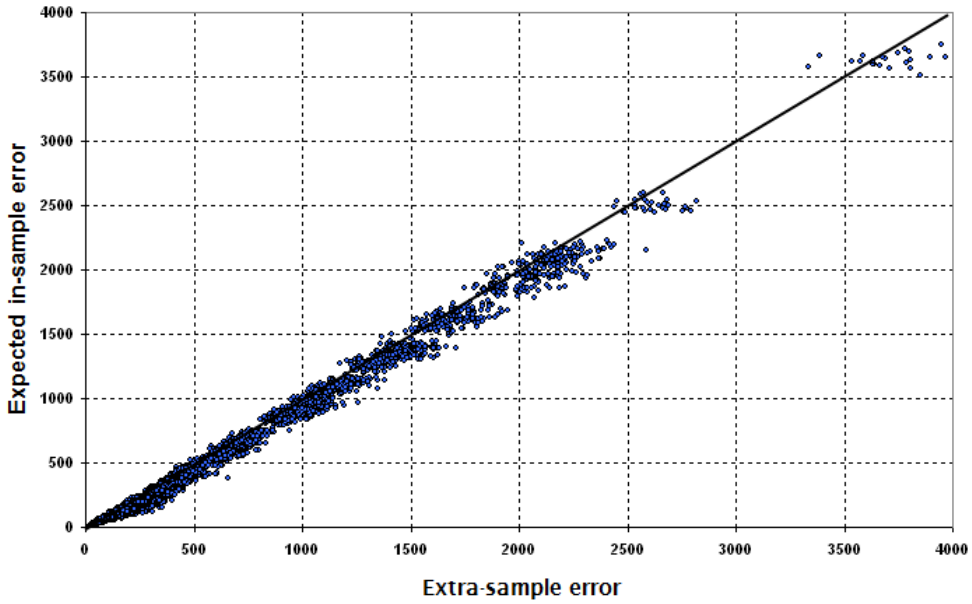

Fig. 2.5 – *Extra-sample error* vs expected *in-sample error*. TREES with n=500, number of samples=30000.

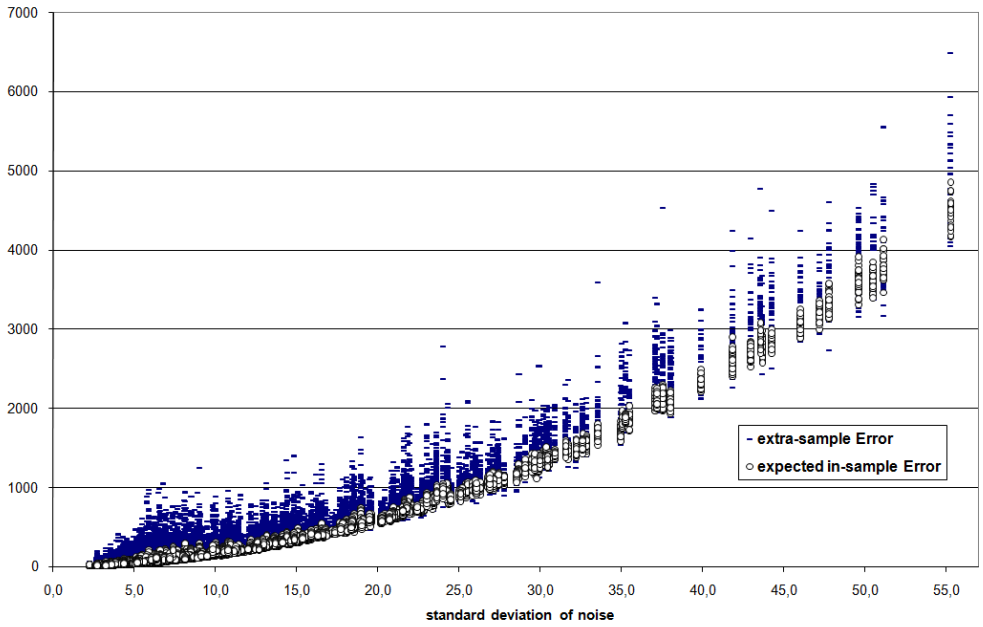

Fig. 2.6 – *Extra-sample error* and expected *in-sample error* vs standard deviation of noise. PPR with n=120, number of samples=30000.

In fig. 2.6 we can see the *extra-sample error* (3) and the expected *in-sample error* (9), computed on 30.000 samples drawn from 1000 generating distributions. The predictive model is the Project Pursuit Regression with sample size 120 (for details see section 9). The X-axis is the standard deviation of the noise, as explained in section 4. Expected *in-sample error* shows a remarkable lower variability than *extra-sample error* particularly for low values of noise standard deviation.

3. Apparent error and optimism

When the data distribution is unknown, we cannot calculate (3), (4) ,(7) or (9), so it is necessary to use an estimator.

The simplest estimator is the *apparent error* (AE) or *resubstitution error*, defined as

$$err = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \hat{f}_{\mathbf{c}}(\mathbf{x}_i)\right) \qquad \mathbf{x}_i \in \mathbf{X}_{\mathbf{c}} \quad y_i \in \mathbf{y}_{\mathbf{c}} \tag{13}$$

i.e. the average of the loss function on the training data-set **c**.

The expression (13) can be calculated easily but it is well known that *err* is an optimistic estimator both of the *extra-sample error Err* and *in-sample error* Err_{in}, because it uses the same data for both the training and evaluation of the model. Moreover, using powerful non-linear models such as Neural Networks, it is possible to obtain a very low value of the *apparent error* just by including more parameters in the model.

The *optimism* is usually defined as the difference between Err_{in} and *err* on the training data, while the *expected optimism* is defined as

$$\overline{op} = E_{\mathbf{Y}_{\mathbf{c}}}\left[Err_{in} - err \mid \mathbf{X}_{\mathbf{c}}\right] \tag{14}$$

that is, the difference between Err_{in} and *err* on new training data, having random *Y* but fixing **X** at the observed sample values (Efron, 1986, Tibshirani & Knight, 1999).

Note that $\hat{f}_{\mathbf{c}}$ in (14) changes for each training sample: $E_{\mathbf{Y_c}}$ averages with respect to training samples which have fixed $\mathbf{X_c}$ (the values of the training set) and different $\mathbf{y_c}$.
Expected optimism is typically positive, since the *apparent error* is usually biased downward as an estimate of the (*in-sample*) *prediction error*.

For a quadratic loss and a general function $f(\mathbf{X})$, from (14) we obtain (Efron, 1986):

$$\overline{Err}_{in} = E_{\mathbf{Y_c}}[Err_{in}] = E_{\mathbf{Y_c}}[err] + \overline{op} = E_{\mathbf{Y_c}}[err] + \frac{2}{n}\sum_i \mathrm{cov}_{\mathbf{Y_c}}(\hat{\mu}_{i\mathbf{c}}, Y_i) \qquad (15)$$

The expression (15) can be obtained by:

$$E_{\mathbf{Y_c}}[Err_{in}] = E_{\mathbf{Y_c}}\left\{\frac{1}{n}\sum_i E_{Y_i}[Y_i - \hat{\mu}_{i\mathbf{c}}]^2\right\} = \frac{1}{n}\sum_i E_{Y_i} E_{\mathbf{Y_c}}[Y_i - \mu_i + \mu_i - \hat{\mu}_{i\mathbf{c}}]^2 =$$

$$= \frac{1}{n}\sum_i E_{Y_i} E_{\mathbf{Y_c}}\left[(Y_i - \mu_i)^2 + (\hat{\mu}_{i\mathbf{c}} - \mu_i)^2 - 2(Y_i - \mu_i)\cdot(\hat{\mu}_{i\mathbf{c}} - \mu_i)\right]^2 =$$

$$= \sigma_\varepsilon^2 + \frac{1}{n}\sum_i E_{\mathbf{Y_c}}(\hat{\mu}_{i\mathbf{c}} - \mu_i)^2$$

$$E_{\mathbf{Y_c}}[err] = E_{\mathbf{Y_c}}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{\mu}_{i\mathbf{c}})^2\right] = \frac{1}{n}\sum_i E_{\mathbf{Y_c}}[Y_i - \mu_i + \mu_i - \hat{\mu}_{i\mathbf{c}}]^2 =$$

$$= \frac{1}{n}\sum_i E_{\mathbf{Y_c}}(Y_i - \mu_i)^2 + \frac{1}{n}\sum_i E_{\mathbf{Y_c}}(\hat{\mu}_{i\mathbf{c}} - \mu_i)^2 - \frac{2}{n}\sum_i E_{\mathbf{Y_c}}[(Y_i - \mu_i)\cdot(\hat{\mu}_{i\mathbf{c}} - \mu_i)] =$$

$$= \sigma_\varepsilon^2 + \frac{1}{n}\sum_i E_{\mathbf{Y_c}}(\hat{\mu}_{i\mathbf{c}} - \mu_i)^2 - \frac{2}{n}\sum_i E_{\mathbf{Y_c}}[(Y_i - \mu_i)\cdot(\hat{\mu}_{i\mathbf{c}} - \mu_i)]$$

and finally

$$\overline{op} = \frac{2}{n}\sum_i E_{\mathbf{Y_c}}[(Y_i - \mu_i)\cdot(\hat{\mu}_{i\mathbf{c}} - \mu_i)] = \frac{2}{n}\sum_i \mathrm{cov}_{\mathbf{Y_c}}(Y_i, \hat{\mu}_{i\mathbf{c}}) \qquad (16)$$

In (15) the covariance penalty term added to the *apparent error* indicates the influence of the values of Y_i on the estimates $\hat{\mu}_{i\mathbf{c}}$ in the sample **c**.

We can note that the covariance penalty theory can be generalized to a broad and important class of loss functions $L(.)$, the Bregman *q*-class divergence. This class accounts for different types of dependent variables and includes the quadratic loss, misclassification loss and other popular loss functions (see Efron, 2004; Zangh, 2008).

The covariance term in (15) and (16) is used by Ye (1998) to define an extension of degree of freedom. Given a modeling procedure *M*, the generalized degrees of freedom (GDF) is defined as:

$$GDF[M] = \sum_{i=1}^{n} h_i^M \quad \text{where} \quad h_i^M = \frac{\partial E_{\mathbf{y_c}}[\hat{\mu}_{i\mathbf{c}}]}{\partial \mu_i} = \frac{1}{\sigma_\varepsilon^2}\mathrm{cov}_{\mathbf{Y_c}}(Y_i, \hat{\mu}_{i\mathbf{c}}). \qquad (17)$$

Therefore GDF is defined as the sum of the average sensitivities of the fitted value $\hat{\mu}_{i\mathbf{c}}$ to a small change in y_i and we can see it as an extension of the trace of the hat matrix (used as degree of freedom for linear smoothers). When the model *M* is very flexible, the fitted values tend to be close to the observed values and therefore the sensitivity of the fitted to the observed values could be high and GDF very large. Ye (1998) proposed a Monte Carlo method to estimate $GDF[M]$ with respect to the derivative formulation:

$$h_i^M = \frac{\partial E_{\mathbf{y_c}}[\hat{\mu}_{i\mathbf{c}}]}{\partial \mu_i} = \lim_{\delta \to 0} E_{\mathbf{y_c}}\left[\frac{\hat{\mu}_i(\mathbf{y_c} + \delta \mathbf{I}_i) - \hat{\mu}_i(\mathbf{y_c})}{\delta}\right] \tag{18}$$

where $\mathbf{I}_i$ is the *i*-th column of the $n \times n$ identity matrix. In other words, an identified structure is stable if the same structure would be identified with perturbed values $\mathbf{y_c} + \boldsymbol{\delta}$ where $\boldsymbol{\delta} \sim N(0, \tau^2 \mathbf{I})$. In this algorithm different values δ_t are generated (t=1,2,….T, where T is the number of replications) from a Gaussian density with variance τ^2 depending on the unknown value of σ_ε^2 and leading to a looping situation. Others authors proposed to estimate directly the covariance terms (see section 7 for the parametric bootstrap approach of Efron) but incurring in the same problem of σ_ε^2 estimation.

4. Estimators based on Cross-Validation.

In K-fold Cross-Validation (CV), we split the sample into K disjoint subsets t_h of (approximately) equal size. We train the model K times, each time leaving out one of the subsets from the training, but using only the omitted subset to estimate the *prediction error*. The mean of these K values is the CV-estimate of the *extra-sample error*.

Indicate with $\mathbf{c}^h$ the training set obtained removing the *h-th* subset t_h and let $m = n/K$ be the number of units in each subset (assuming that n is a multiple of K). The CV-estimator is defined as the average error on the K analyses:

$$err^{CV} = \frac{1}{K}\sum_{h=1}^{K}\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} L\left(y_j, \hat{f}_{\mathbf{c}^h}(\mathbf{x}_j)\right) \tag{19}$$

To simplify the notation we have indicated by $j \in t_h$ the expression $(y_j; \mathbf{x}_j) \in t_h$.

It is known that k-fold CV is a biased estimate of *Err* and that by increasing the number of folds we can reduce the bias (Kohavi, 1995). This is due to the fact that err^{CV} is based on functions $\hat{f}_{\mathbf{c}^h}$ estimated on samples of size (*n-m*), so it tends to overestimate *Err*.

We can interpret expression (19) also from another point of view. If we see the training sets $\mathbf{c}^1, \mathbf{c}^2, ..., \mathbf{c}^K$ as samples of size (*n-m*), we are estimating $\hat{f}_{\mathbf{c}^h}$ on different samples, so err^{CV} is an unbiased estimator of the expectation of Err for sample-size (*n-m*). We can thus obtain an approximate estimation of $E_{\mathbf{c}}(Err) \approx E_{c^h}\left(Err \mid \hat{f}_{\mathbf{c}^h}, c^h\right)$.

The main problem with K-fold CV is that the training-sets $\mathbf{c}^1, \mathbf{c}^2, ..., \mathbf{c}^K$ are not independent samples, i.e. they have $(n - 2m)$ cases in common, and also the test sets t_h come from the same data. This implies that the variance of err^{CV} may be very large (Breiman, 1996) and several authors (Dietterich, 1998; Bengio & Grandvalet, 2003) have considered the difficulties of the variance estimation showing that no unbiased estimator of $Var\left(err^{CV}\right)$ can be obtained.

Note that err^{CV} defined by (19), using a quadratic loss function, can be written as the average of identically distributed (dependent) variables:

$$err^{CV} = \frac{1}{K}\sum_{h=1}^{K}\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} \delta_j \tag{20}$$

with $\delta_j = \left(y_j - \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_j\right)\right)^2$, for $j \in t_h$.

Then err^{CV} asymptotically converges to a normally distributed variable with variance (Bengio & Grandvalet, 2004):

$$Var\left(err^{CV}\right) = \frac{1}{n^2}\sum_{ij} Cov\left(\delta_i, \delta_j\right) = \frac{1}{n}\sigma_\delta^2 + \frac{m-1}{n}\omega + \frac{n-m}{n}\gamma \qquad (21)$$

Where

- σ_δ^2 is the average variance (taken over training sets) of δ_j for true test cases (i.e. sampled independently from the training sets), considering training samples of size $m(k-1)$;
- ω is the covariance of δ_j for the same "true" test set t_h, i.e. it measures the dependence of test errors stemming from the common training set;
- γ is the covariance of δ_j for different test sets, i.e. it measures the dependence of the training sets (each couple of training sets share $m(k-2)$ cases) and the dependence due to the fact that the test-set cases of t_h are included in each training set $\mathbf{c}^j$ with $j \neq h$.

Expression (21) is the sum of $n + K\left[\frac{m(m-1)}{2}\right] + \left(\frac{n^2 - Km^2}{2}\right) = \frac{n(n+1)}{2}$ covariance terms, but there are only three different values.

If K equals the training sample size n, we obtain the "Leave-One-Out" Cross-Validation (LOO). So LOO can be considered a particular case of K-fold Cross-Validation.
Note that LOO is known to fail to estimate $E_{\mathbf{c}}(Err)$ for unsmooth statistics (Breiman, 1996; Efron & Tibshirani, 1993). This failure is due to the similarity of the training sets $\mathbf{c}^1, \mathbf{c}^2, ..., \mathbf{c}^n$ which are far from being independent samples of dimension (n-1).

In particular LOO is an almost unbiased estimator of *Err* but it has high variability producing non-reliable estimates (Efron, 1983; Stone, 1977). Furthermore it was showed that using LOO for model selection, we obtain an inconsistent procedure in the sense that the probability of selecting the model with the best predictive ability does not converge to 1 as the size of the dataset tends to infinity. On the contrary, K-fold CV shows lower variability than LOO and it results consistent in linear model selection problems but it may have large bias (Shao, 1993; Zhang, 1993).
Some authors defined sanity-check bounds which show that the LOO estimator performance will not be never considerably worse than the *apparent error* estimator, under a weak assumptions of error stability (Kearns & Ron, 1999). Computation of LOO is heavy also for moderate sample size, but when we are considering a linear regression (or a linear smoother), it is possible to use the Generalized Cross Validation estimator (Hastie et al., 2001).

Many simulation and empirical studies have verified that a reliable estimate of *Err* can be obtained with K=10 for n>100 as recommended by Davison et al. (1997). In choosing subsets of inputs in linear regression, Breiman and Spector (1992) found 10-fold and 5-fold cross-validation to work better than Leave-one-out. The variance of LOO is generally larger than 10-fold CV and this could be due to the instability of the model used as, for example, in the case of decision trees (Kohavi, 1995; Elisseeff & Pontil, 2003).

Burman (1989) showed that $E\left(err^{CV} - Err\right) \approx c_0 (K-1)^{-1} n^{-1}$. For K=n the right-side term of the expression is $O\left(n^{-2}\right)$, but when K is small this term is not necessarily very small. Moreover, the constant term c_0 is of the same order of the number of parameters being estimated and so CV may give a poor estimate of Err if the number of parameters is large. Therefore, Burman introduced a corrected version of cross-validation, CV*:

$$err^{CV*} = err^{CV} + err - \bar{e}^{+} \tag{22}$$

where $\bar{e}^{+} = \frac{1}{K}\sum_{j=1}^{K} e_j^{+}$ and e_j^{+} are obtained considering K-1 folds for model estimation and, differently from CV, the whole sample for testing.

We can write expression (22) as

$$err^{CV*} = \frac{1}{K}\sum_{h=1}^{K}\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} L\left(y_j, \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_j\right)\right) + \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \hat{f}_{\mathbf{c}}\left(\mathbf{x}_i\right)\right) - \frac{1}{K}\sum_{h=1}^{K}\frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_i\right)\right) =$$

$$= \frac{1}{K}\sum_{\mathrm{h}=1}\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} L\left(y_j, \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_j\right)\right) + \frac{1}{n}\sum_{\mathrm{i}=1}\left[L\left(y_i, \hat{f}_{\mathbf{c}}\left(\mathbf{x}_i\right)\right) - \frac{1}{K}\sum_{\mathrm{h}=1} L\left(y_i, \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_i\right)\right)\right] \tag{23}$$

The second term of (23) will usually be negative to compensate for the bias of err^{CV}.

The author showed that $E\left(err^{CV*} - Err\right) \approx c_1 (K-1)^{-1} n^{-2}$ where the constant c_1 depends on the loss function $L(\cdot)$ and on the density distribution of the variables.

In short, err uses a function $\hat{f}_{\mathbf{c}}$ estimated on all data, while e_j^{+} uses functions estimated on (*n*-*m*) data. Consequently, usually, $err < e_j^{+}$ and the difference will be greater the larger the amount of overfitting, and thus, the larger the value of err^{CV}.

We can also write expression (22) as:

$$err^{CV*} = err + (err^{CV} - \bar{e}^{+}) \tag{24}$$

where $(err^{CV} - \bar{e}^{+})$ can be seen as an estimate of *optimism* referring to *extra sample error*.

Let $\delta_{jh} = L\left(y_j, \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_j\right)\right)$. We can write, differently from (14), this estimate of *optimism* as:

$$\hat{op}_E = \frac{1}{K}\sum_{h=1}^{K}\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} L\left(y_j, \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_j\right)\right) - \frac{1}{K}\sum_{h=1}^{K}\frac{1}{n}\sum_{j=1}^{n} L\left(y_j, \hat{f}_{\mathbf{c}^h}\left(\mathbf{x}_i\right)\right) =$$

$$= \frac{1}{K}\sum_{h=1}^{K}\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} \delta_{jh} - \frac{1}{K}\sum_{h=1}^{K}\frac{1}{n}\sum_{j=1}^{n} \delta_{jh} = \frac{1}{K}\sum_{h=1}^{K}\left[\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} \delta_{jh} - \frac{1}{n}\sum_{j=1}^{n} \delta_{jh}\right] =$$

$$= \frac{1}{K}\sum_{h=1}^{K}\left[\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} \delta_{jh} - \frac{1}{n}\sum_{\mathrm{j}\in \mathrm{t_h}} \delta_{jh} - \frac{1}{n}\sum_{\mathrm{j}\notin \mathrm{t_h}} \delta_{jh}\right] = \frac{1}{K}\sum_{h=1}^{K}\left[\frac{n-m}{mn}\sum_{\mathrm{j}\in \mathrm{t_h}} \delta_{jh} - \frac{m}{mn}\sum_{\mathrm{j}\notin \mathrm{t_h}} \delta_{jh}\right] =$$

$$= \frac{n-m}{n}\left[\frac{1}{K}\sum_{h}\frac{1}{m}\sum_{\mathrm{j}\in \mathrm{t_h}} \delta_{jh} - \frac{1}{K}\sum_{h}\frac{1}{n-m}\sum_{\mathrm{j}\notin \mathrm{t_h}} \delta_{jh}\right] = \frac{K-1}{K}\left[err^{CV} - \frac{1}{K}\sum_{h} err_h^{n-m}\right] =$$

$$= \frac{K-1}{K}\left[err^{CV} - \overline{err}^{n-m}\right] \tag{25}$$

The first term in the square brackets is the mean function's fit to test sets, i.e. err^{CV} , the second term is the mean function's fit to the training sets, i.e. the average *apparent error* on (*n*-*m*) data.

The Hold-out estimator (HO) is similar to a cross-validation estimator. It is obtained splitting randomly the sample in a training set to estimate the model and a test set to evaluate the prediction error. If we split the sample into two subsamples of equal size, repeating two time the procedure swapping test data with training data, we obtain an estimator very similar to 2-fold cross-validation. For a classification problem, Kearn (1997) gives a general bound on the error of the hold-out estimator considering the approximation rate (the accuracy to which the true function can be approximated as a function of the number of model parameters). The author shows that the bound is strictly linked to the fraction of sample used for testing. When the complexity of the true function is small compared to the sample size, the performance of the hold-out estimator is relatively insensitive to the choice of the fraction of cases used for the test set. Therefore there is a fixed value of the fraction which seems to yield reasonably good performance for a wide range of values of complexity. Surprisingly the optimal value of the fraction tends to one as the size of the sample tends to infinity, indicating that most data should be used for testing. Usually, the dimension of the training set is 2/3 of the sample.

5. Repeated cross-validation estimators

The values obtained by HO and, at a lower level, CV depend on the initial random partition of the sample. A method to reduce this dependence obtaining a more reliable estimate, consists in repeating the procedure a small number of times with a different random split. The average value obtained is the Repeated CV (RCV) or the Repeated Corrected CV (RCV*) (Burman, 1989; Molinaro et al., 2005), or the Repeated Hold-Out (RHO), also named Monte Carlo Cross-Validation (Dudoit & Van der Laan, 2005).
Another cross-validation technique is the *complete* K-fold CV or leave-*m*-out CV (Shao, 1993). Indicating with $m = \frac{n}{K}$ the test set size in a K-fold CV, we can consider the average of all different values obtained splitting in $\binom{n}{m}$ ways the data set. However, when *m* is large, the amount of computation required to use the *complete* cross-validation may be impractical.

Several authors have applied RCV to concrete problems, but there are not papers showing the real advantage of this approach (except for Burman, 1989).
To evaluate the influence of repetition on cross-validation estimators we can consider the simulation showed in fig. 5.1. In this case we can see what happens for one generating distribution, using Regression Trees and sample size 120. We have drawn 500 samples from one distribution (obtained as described in section 9) and for each sample we have calculated: *Err* (on a 50.000 cases test-set), 100 (random split) CV and one RCV (the mean of the 100 random CV).
The obtained average $\overline{Err}$ was 35.84 with standard deviation 2.0.
CV and RCV have mean 37.62, but MSE of CV is 25.75 and MSE of RCV is 20.77. That is, Repeated CV shows an important reduction of variability.

In the figure we can see the average $\overline{Err}$ (the horizontal red line), the single CV estimates (blue points) and the RCV estimate (yellow points). The black line represents the values of *Err*, that is the values we want to estimate.

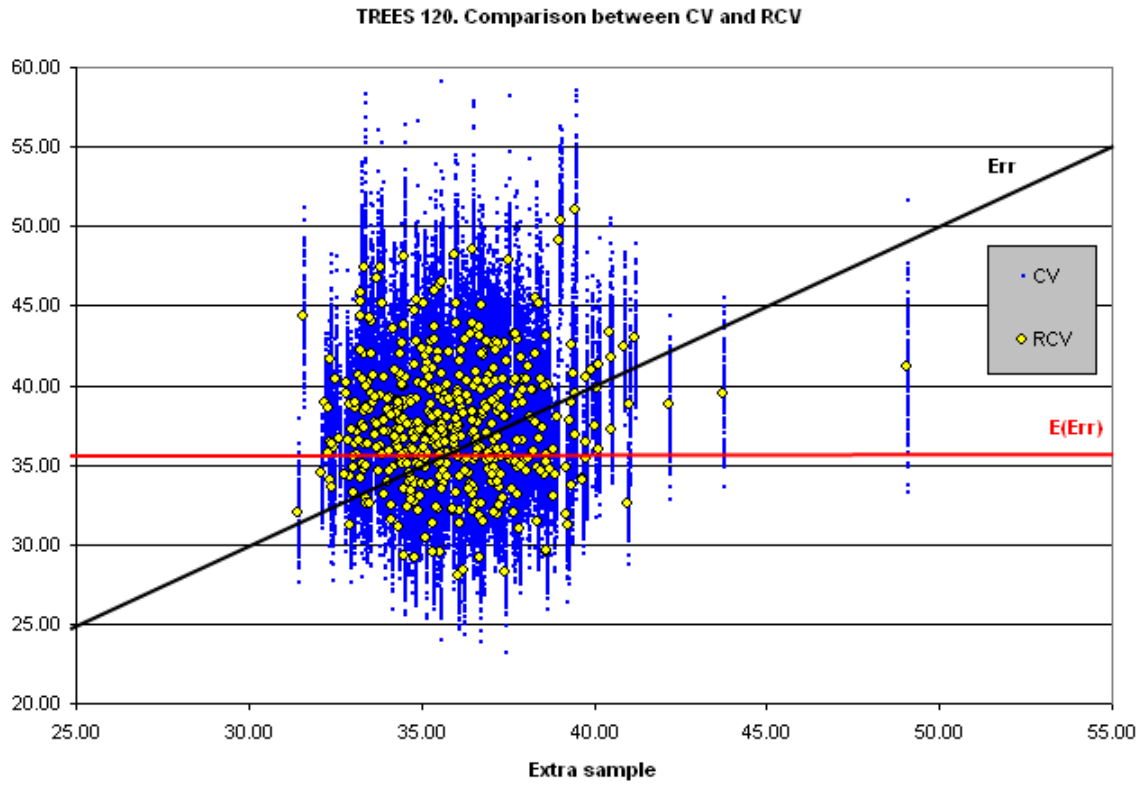

Fig. 5.1 - Comparison between CV and RCV(100 random splits) for Regression Trees: 500 samples of size 120 from one data generating distribution, with respect to *extra-sample error*.

6. Estimators based on non-parametric Bootstrap

Instead of repeatedly analyzing subsets of the data as in RHO, it is possible to analyze bootstrap samples of data. Given a collection of K samples, $\{b_1, b_2, \ldots, b_K\}$ of size *n*, drawn with replacements from the training set, the model is estimated on each sample b_j, and the estimated *prediction error* for the b_j sample, err_j^{Bt}, is calculated considering as a test sample the cases not included in b_j.

The *Leave-one-out bootstrap* estimator of *Err* is defined as:

$$err^{Bt} = \frac{1}{K}\sum_{j=1}^{K} err_j^{Bt} \qquad (26)$$

The properties of this estimator, compared with Cross-Validation and Hold-out estimators, have been analysed in many theoretical and empirical studies (Efron & Tibshirani, 1997). This estimator is biased upwards, but is considered to have lower variability with respect to Cross-validation or Hold-out estimators.

To improve err^{Bt}, Efron (1983) proposed the .632 bootstrap estimator:

$$err^{B632} = 0.632 \cdot err^{Bt} + 0.368 \cdot err \tag{27}$$

designed to correct the upward bias in err^{Bt} by averaging it with the downwardly biased estimator err. The weights are based on the fact that bootstrap samples include approximately 0.632*n* cases of the training data. However, in all situations of severe overfit, the estimator err^{B632} is downwardly biased because $err = 0$. To avoid this problem, Efron and Tibshirani (1997) proposed a new estimator, the .632+ bootstrap estimator, err^{B632+}, designed to be a less-biased compromise between err and err^{Bt}. It assigns greater weight to err^{Bt} when the amount of overfitting is large.

To calculate err^{B632+}, firstly the *no-information error*, γ, is introduced corresponding to the expected *prediction error* of model $\hat{f}(\mathbf{X})$ when Y and $\mathbf{X}$ are independent. Given $\hat{f}(\mathbf{X})$, an estimate of γ is obtained considering the loss on all n^2 couples $(y_i, \mathbf{x}_j)$:

$$\hat{\gamma} = \frac{1}{n^2}\sum_{i,j} L\left(y_i, \hat{f}\left(\mathbf{x}_j\right)\right) \qquad (i,j=1,2,\ldots,n) \tag{28}$$

Thus a relative overfitting rate is defined as:

$$\hat{R} = \left(err^{Bt} - err\right)/\left(\hat{\gamma} - err\right) \tag{29}$$

with its range forced to be between 0 (no overfitting) and 1 (high overfitting).

The .632+ bootstrap estimator is given by:

$$err^{B632+} = \hat{w} \cdot err^{Bt} + \left(1 - \hat{w}\right) \cdot err \tag{30}$$

where $\hat{w} = 0.632/(1 - 0.368 \cdot \hat{R})$. The weight ŵ ranges from 0.632 (no-overfitting) to 1 (severe overfitting).

7. Estimators based on parametric Bootstrap and covariance penalties

A seen in section 3 a way to evaluate the prediction capability of the model is to estimate the *optimism* and then add it to the *apparent error*. Many methods based on this idea are been proposed, specially for the modeling selection process, as the well known AIC, BIC or Mallows' C_p estimators.

Expression (15) can be seen as a generalization of Mallows' C_p for a general estimator of $f(\mathbf{X})$. In fact, considering a linear rule $\boldsymbol{\mu} = \mathbf{M}\mathbf{y}$ with hat matrix $\mathbf{M}$ and $\mathbf{y}$ generated from a homoscedastic Gaussian model, the covariance penalty term $\sum_i \mathrm{cov}_{\mathbf{y}}\left(y_i, \hat{\mu}_{i\mathbf{c}}\right)$ is equal to $\sigma_\varepsilon^2 trace(\mathbf{M})$ and, for a linear regression model with p parameters, to $\sigma_\varepsilon^2 p$. We must remark that expression (15) considers the expected *in-sample error* while CV and Nonparametric Bootstrap are estimators of the *extra-sample error*.

As shown in fig. 7.1, the Parametric boostrap for the small sample understimates the *extra-sample error*, but with a larger sample and a more stable model the estimator becomes quite unbiased (see fig. 7.2).

In the literature, several methods have been proposed to estimate the *optimism*. Considering a nonlinear estimation rule $\hat{\boldsymbol{\mu}} = g(\mathbf{y})$, if $g(.)$ is a smoother function for a homoskedastic Gaussian model, under differentiability conditions, it is possible to use Stein's unbiased risk estimate (Stein 1981):

$$err^{SURE} = err + \frac{2}{n}\sum_i \frac{\partial \hat{\mu}_i}{\partial y_i} \tag{31}$$

Other authors proposed to estimate the covariance term by bootstrap. Recently, Efron (2004) introduced a parametric bootstrap procedure consisting of the following steps:

1. Estimate $\hat{\boldsymbol{\mu}} = g(\mathbf{y})$, while $\hat{\sigma}^2$ can be obtained applying a model sufficiently complex to have negligible bias.
2. From the bootstrap density $\hat{\mathbf{f}} = N(\hat{\boldsymbol{\mu}}, \hat{\sigma}^2\mathbf{I})$ generate, for each $\mathbf{x}_i$ of the data-set, B new values y_i^{*b} and estimate $\hat{\mu}_i^{*b} = g(y_i^{*b})$.
3. Estimate $\mathrm{cov}_{\mathbf{y}}(y_i, \hat{\mu}_i)$ by

$$\hat{\mathrm{cov}}_i = \sum_{b=1}^{B} \hat{\mu}_i^{*b}\left(y_i^{*b} - \bar{y}_i^{*}\right) \Big/ (B-1) \qquad \text{with} \qquad \bar{y}_i^{*} = \sum_b \frac{y_i^{*b}}{B} \tag{32}$$

4. Finally, the estimator is given by:

$$err^{PB} = err + \frac{2}{n}\sum_i \hat{\mathrm{cov}}_i \tag{33}$$

To adapt Efron's proposal of Parametric Bootstrap to a complex non-parametric model (e.g. Neural Networks), we adopted a two-step procedure.

In the first step we need to obtain good estimates of σ_ε and $\boldsymbol{\mu}$. In fact, Efron's proposal to use a "big" model is not applicable when considering models capable of fitting random noise: in this case, it is possible to obtain $\hat{\boldsymbol{\mu}} = \mathbf{y}_{\mathbf{c}}$ and $\hat{\sigma}_\varepsilon = 0$.

To overcome this problem, in the first step of our simulation we applied a non-parametric bootstrap procedure in order to obtain robust estimates of σ_ε and $\boldsymbol{\mu}$. Once we have obtained these estimates, we can define the bootstrap density $\hat{\mathbf{f}} = N(\hat{\boldsymbol{\mu}}, \hat{\sigma}^2\mathbf{I})$ and then calculate formula (33). This estimation is illustrated in the following Matlab code, which uses a PPR model:

```
% niter = n. of Bootstrap iterations
% n = number of units
        sigma = 0.0;
        mu    = zeros(n,1);
   for i=1:niter
       % generate from YX one nonparametric bootstrap sample YXb
        [Yb Xb] = bootstrap(Y,X);
       % estimate Y by Projection Pursuit Regression on Yb Xb
        smod = ppreg(Yb,Xb,min,max);
        Yhat = pphat(X,smod) ;
        ydif(:,i) = Y - Yhat ;
        sigma   = sigma + std(ydif(:,i));
        mu = mu + Yhat;
   end;
% sigma = mean of standard deviations in each samples
        sigma = sigma/niter;
% mu = mean prediction vector for each sample
        mu    = mu/niter;
```

Zhang (2008) proposed an alternative model-free approach to estimate the covariance-penalty based on Leave-one-out. The author suggested an approximated LOO formulas which facilitate fast estimation of prediction error when the conditional distribution of the response variable Y belong to the exponential family and the prediction error is based on the Bregman divergence measure.
In a classification context, Daudin and Mary-Huard (2008) presented a similar method for prediction error estimation based on the covariance terms and named swapping method. The

authors showed, by using artificial and real data sets, that their method is very competitive compared to CV using the K-nearest-neighbor classificator.

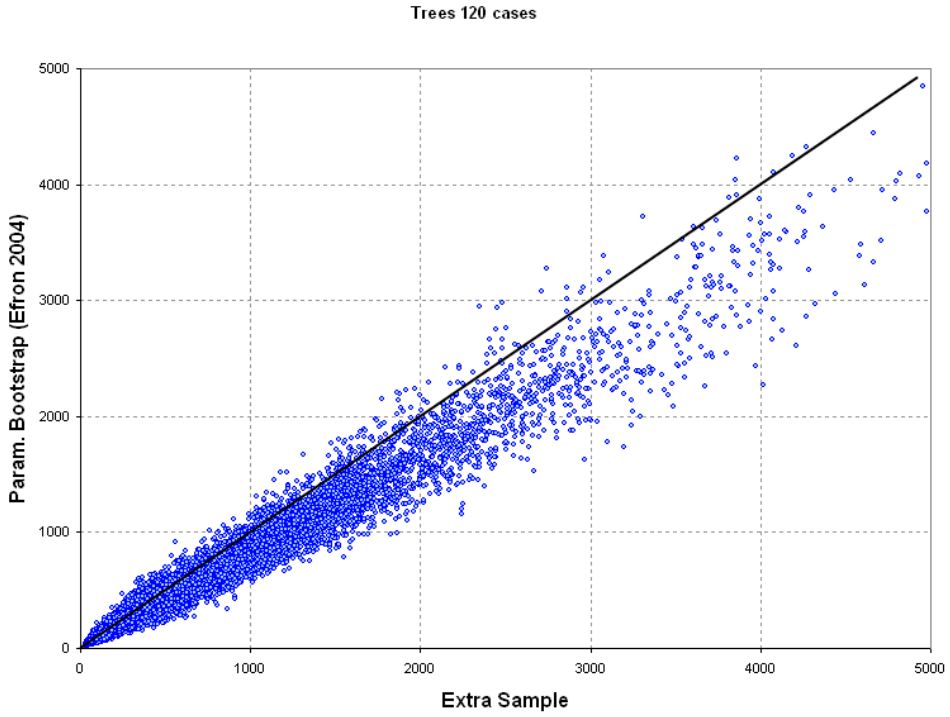

Fig. 7.1 – Parametric bootstrap vs *extra-sample error*. TREES 120, 30000 samples

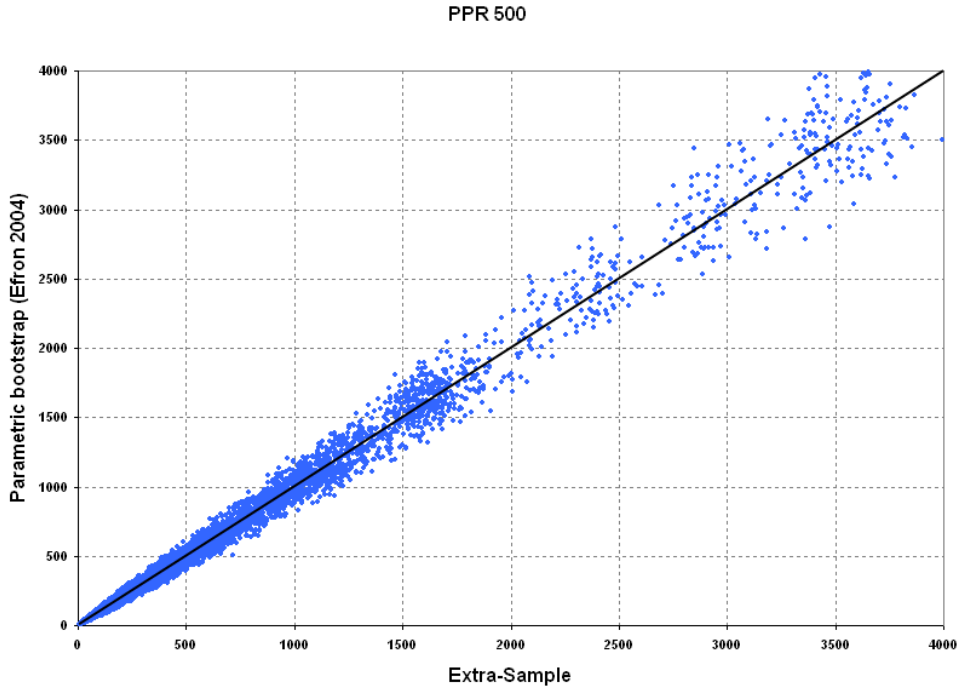

Fig. 7.2 – Parametric bootstrap vs *extra-sample error*, PPR 500, 30000 samples.

8. A unified view of the estimators

It is possible to view several of the previous estimators as a particular case of a general CV estimator (Dudoit & Van der Laan, 2005).
Introduce a binary split vector $\boldsymbol{\delta}$ defining a particular split of the sample into a training-set c_δ and a test-set t_δ ($\delta_i = 0 \Rightarrow$ *i-th unit* in the training-set). Let p_t be the proportion of observations in the test set, $p_t = \sum \delta_i / n$.
We can generalize the expression of the CV estimator as:

$$err^{GCV} = E_\delta \underset{(y,\mathbf{x})\in t_\delta}{E} \left[L\left(y, \hat{f}_{c_\delta}(\mathbf{x})\right) \right] = E_\delta \left[\frac{1}{n_t} \sum_{(i:\delta_i=1)} L\left(y_i, \hat{f}_{c_\delta}(\mathbf{x}_i)\right) \right] \tag{34}$$

If we consider K split vectors ${}^1\boldsymbol{\delta}$, ${}^2\boldsymbol{\delta}$, ${}^3\boldsymbol{\delta}$,... ${}^K\boldsymbol{\delta}$, defining K mutually exclusive and exhaustive test-set of approximately equal size, we obtain the K-fold CV defined in (19), having $p_t = 1/K$. If $n_t = 1$, we obtain LOO and $p_t = 1/n$.
Drawing randomly the split vectors, having fixed p_t (usually 1/3), we obtain the Repeated Hold Out estimator (RHO).
Defining bootstrap inside this scheme is a bit more complicated.
Let be ${}^1\boldsymbol{\varsigma}$, ${}^2\boldsymbol{\varsigma}$, ${}^3\boldsymbol{\varsigma}$,... ${}^r\boldsymbol{\varsigma}$ a set of random binary vectors such that

$$\sum_j \left(n - \sum_i {}^j\varsigma_i \right) = rn - \sum_j \sum_i {}^j\varsigma_i = n \tag{35}$$

Now define
$\delta_i = 1$ *if* ${}^j\varsigma_i = 1$, $\forall j$
$\delta_i = 0$ *otherwise*
where the training units have the weights $w_i = r - \sum_j {}^j\varsigma_i$.
In this case p_t is a random variable, but we know (see section 6) that $E(p_t) \approx 0.368$.
We have then obtained the Leave-One-Out Bootstrap estimator (26). Anyway, we cannot derive the more interesting B632 or the B632+ estimators, the corrected CV or the parametric Bootstrap of Efron, so this unified view has not great interest for this study.

9. The simulation framework

Given five variables Y, X_1, X_2, X_3, X_4 we considered 1000 data generating distributions, in which $Y = f(X_1, X_2, X_3, X_4) + \varepsilon$ and f is the non-linear function

$$f(\mathbf{X}) = a\left(c_0 + \sum_{j=1}^4 c_j X_j\right) + \sum_{j=1}^4 \beta_j \left(X_j - \overline{X}_j\right)^2 \tag{36}$$

where

- X_j (j=1,…,*4*) is the j-th explanatory variable generated from a Beta distribution with parameters (g_j, t_j) , where the latter are drawn from a Uniform distribution in the interval [2 , 10];
- a, c_0, c_j and β_j (j=1,…,*4*) are randomly drawn from Uniform distributions in the intervals [0, 4], [-5 , 5], [-5 , 5] and [-50 , 50], respectively;
- ε is the noise generated from a Gaussian distribution $N(0, \sigma_\varepsilon^2)$.

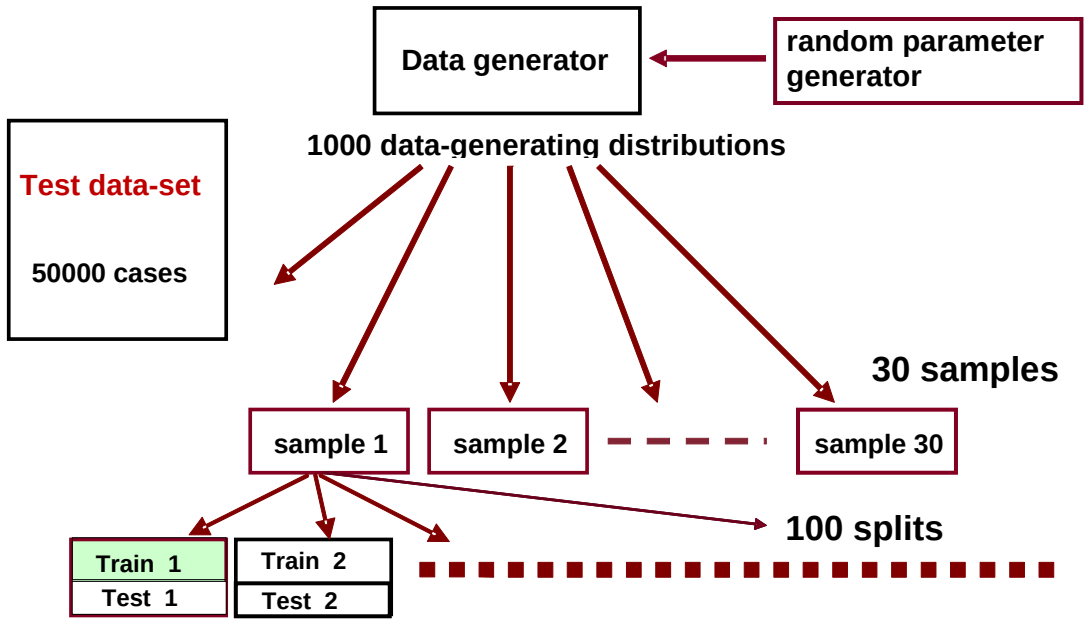

Fig. 9.1 – The simulation scheme for Repeated Hold-Out estimator.

In this diagram we show the main steps needed to carry out the simulation with the Repeated Hold-Out estimator. For the other estimators the simulation change only in the last step: we don't split, repeatedly, the sample in two subsamples, rather we create bootstrap samples or k-fold splits.

The above definitions allow us to generate data with different levels of non-linearity in the relation between Y and $\boldsymbol{X}$. To allow great generality in the results, the systematic component of the model (the signal) has been scaled so that its standard deviation, σ_s, is equal to a random value drawn from a Uniform distribution in the interval [10 , 30]. We then drew randomly the signal-to-noise ratio $r_{s/n}$ (between 0.5 and 4.5). The standard deviation of the noise, σ_ε, was then fixed to be $\sigma_\varepsilon = \sigma_s / r_{s/n}$.

Note that S/n=0.5 means that the true underlying function accounts for 20% of the variance of the response, while S/n=4.5 implies a percentage of 95%.

Several simulations were carried out as follows. One thousand data generating distributions were obtained first drawing the values of a, c_0, c_j, β_j, g_j , t_j , σ_s and $r_{s/n}$; then, conditionally on these, creating the test-set drawing 50000 values for each X_j (j=1, … , 4) and for ε, and finally computing Y.

For each data generating distribution, in the same way as the test-set, we extracted 30 independent samples of size 120 or 500, depending on the simulation (see the next paragraph).

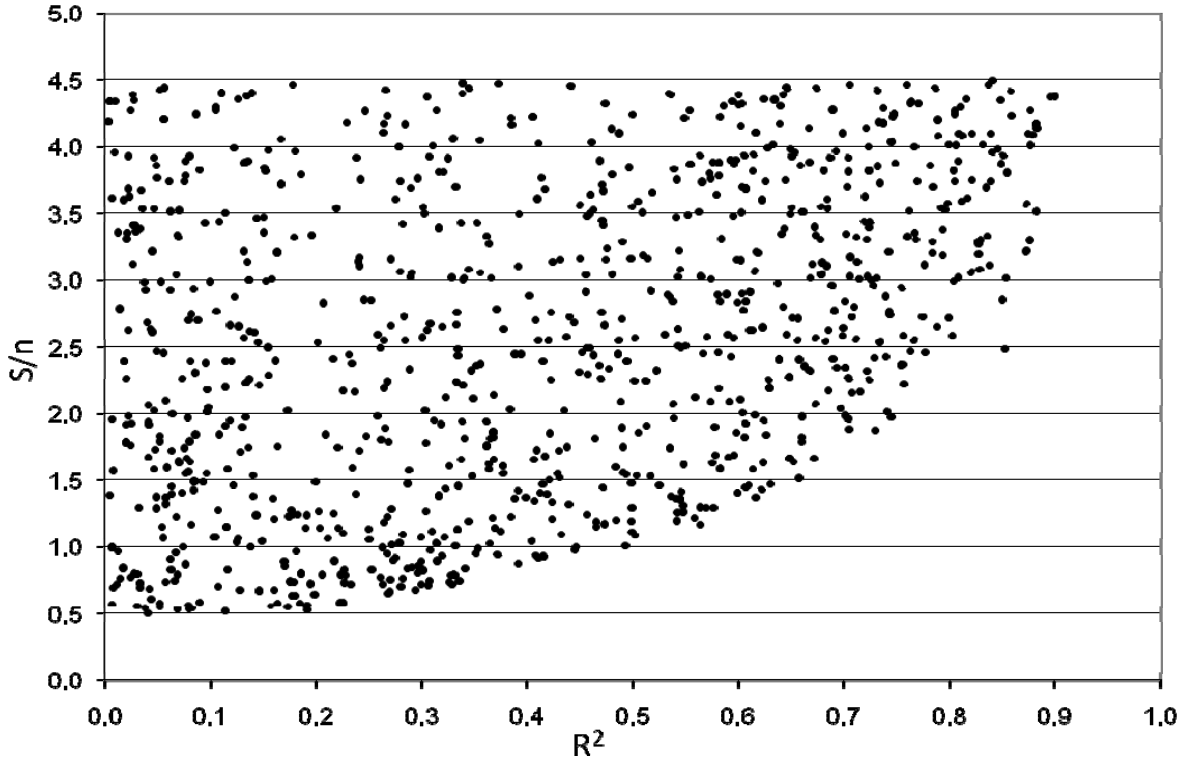

Fig. 9.2 - Plot of 1000 data generating distributions with respect to S/n ratio and R^2

In the fig. 9.2, they are shown 1000 generating distribution by level of linearity (measured by R^2) and noise to signal ratio (S/n). Of course, there can not be generating distributions with small S/n and high R^2. Moreover, R^2 can not be greater than 0.95 (corresponding to S/n=4.5 and a perfectly linear function).

To estimate the function f we chose three different methods:

- The Regression Tree model (RT) as implemented in CART (Breiman et al., 1984). It can fit non-linear data, it is very fast, it is easy to interpret. Moreover it is quite unstable and, having a large number of leafs, can overfit data.
  The Regression Tree model can be defined
  $$Y = \sum_{m=1}^{M} c_m \mathrm{I}(\mathbf{x} \in R_m) + \varepsilon \tag{37}$$
  where $c_m = E(Y \mid \mathbf{x} \in R_m)$ and R_m is one of the M leafs of the regression tree. To control the tree size, we didn't use pruning but we fixed the minimum split size to 10, for sample size 120, and 20, for sample size 500.

- The Projection Pursuit Regression model (PPR), as proposed by Friedman & Stuetzle (1981). We used the algorithm SMART (Friedman, 1985). Given a sufficient number of parameters, PPR can approximate any continuous function, but, for moderate M, it generates quite stable and smooth functions.
  In the PPR we have the following model:
  $$Y = \sum_{m=1}^{M} g_m(\boldsymbol{\beta}'_m \mathbf{x}) + \varepsilon \tag{38}$$
  That is an additive model in the transformed variables $z_m = \boldsymbol{\beta}'_m \mathbf{x}$, where the functions g_m are estimated by nonparametric scatterplot smoother. We fixed M=3, as PPR showed to be an adequate performance in a pre-simulation analysis.

- The Neural Network model (NEU). We considered a simple feedforward Neural Network with one hidden layer with 6 nodes which appeared to be adequate for our data. It is important to note that this Neural Network model has an expression similar to (38) but with simpler functions than g_m (Hastie et al., 2001). Using also an early stopping rule based on a validation set, we expect NEU to be more regular than PPR. For this model we considered only 200 generating distributions.

The loss function considered in this simulation is the usual squared error function. Given a data generating distribution, we carried out the following steps for each sample:

1. Estimation of the models TREE, PPR and NEU on all the units of the sample.
2. Estimation of the predicted values of Y on the large test-set: comparing the estimates with the true values of Y, we obtain, with a small approximation, the *extra-sample error* Err .
3. Estimation of *in-sample error* Err_{in}: generate 300 new response vectors by $N(\boldsymbol{\mu}, \sigma_\varepsilon^2 \mathbf{I})$, with $\mu = f(X_1, X_2, X_3, X_4)$, then compute expression (7).
4. Calculation of all the estimators described in the previous paragraphs, using only the sample.

At step 2), the value of Err was obtained estimating the model on the sample and then calculating the prediction error on the large (50.000 cases) test set. The more complex calculation of Err_{in} is showed in the following Matlab code:

```
% computation of ERRin
    % nrep = number of replications;
    % deve = standard deviation of noise
    ERRin=0.0;
    mu = Y-eps; % mu = E(Y/X)= Y without error
    Ys = zeros(n,nrep);
    % generate nrep new vectors Y/X
    for j=1:nrep
         Ys(:,j) = normrnd(mu,deve);
    end;
    % calculate ERRin as mean of (nrep*(nrep-1)) values
    smod = ppreg(Ys(:,j),X,min,max);
    Yhat = pphat(X,smod) ;
    for j2 = 1:nrep
          ydif = Yhat-Ys(:,j2);
          ERRin = ERRin + (ydif'*ydif /n) ;
    end;
    ERRin = ERRin / double(nrep);
```

Calculation of $\overline{Err}_{in}$ is more demanding and it is showed in the following Matlab code:

```
% computation of expected ERRin
    % nrep = number of replications;
    % deve = standard deviation of noise
    EERRin=0.0;
    mu = Y-eps; % mu = E(Y/X)= Y without error
    Ys = zeros(n,nrep);
    % generate nrep new vectors Y/X
    for j=1:nrep
         Ys(:,j) = normrnd(mu,deve);
    end;
    % calculate EERRin as mean of (nrep*(nrep-1)) values
    for j = 1:nrep
      smod = ppreg(Ys(:,j),X,min,max);
      Yhat = pphat(X,smod) ;
      for j2 = 1:nrep
          if (j ~= j2)
          ydif = Yhat-Ys(:,j2);
          EERRin = EERRin + (ydif'*ydif /n) ;
          end;
      end;
    end;
    EERRin = EERRin / double(nrep*(nrep-1));
```

To account for the sample variability, in a given h-th distribution, we computed the value:

$$\nu_h = \sqrt{\frac{1}{K}\sum_{j=1}^{K}\left(\hat{Err}_{jh} - Err_{jh}\right)^2} \tag{39}$$

where K=30, the number of samples.
This estimator will be zero if $\hat{Err}_j = Err_j$ for each sample, i.e. the estimation procedure gives always the exact value we would obtain (the *extra-sample error* of that model on the generating distribution).
We calculated also the mean of the Err_j in a fixed distribution by:

$$\overline{Err}_h = \frac{1}{K}\sum_{j=1}^{K} Err_{jh} \tag{40}$$

Then we have

$$\varphi_h = std\left(Err_{jh} \mid h\right) = \sqrt{\frac{1}{K}\sum_{j=1}^{K}\left(Err_{jh} - \overline{Err}_h\right)^2} \tag{41}$$

In most cases we should obtain $\nu_h \geq \varphi_h$.
An overall measure of performance on all distributions can be obtained by

$$rse_h = \frac{\nu_h}{\varphi_h} \qquad \text{(relative root squared error for } h\text{-}th \text{ distrib.)} \tag{42}$$

$$\overline{rse} = \frac{1}{H}\sum_{h=1}^{H}\frac{\nu_h}{\varphi_h} \qquad \text{(mean relative root squared error)} \tag{43}$$

If the estimator is unbiased and has low variability, then, for each distribution, rse_h will be small. $\overline{rse}$ considers jointly the main characteristics of an estimator evaluated on a number of different distributions.
As summary estimators of biaseness of the models on all the distributions, we computed the following indices:

$$rb_h = \frac{1}{K}\sum_{j=1}^{K}\frac{E\hat{r}r_{jh} - Err_{jh}}{E\hat{r}r_{jh} + Err_{jh}} \qquad \text{(relative bias for } h\text{-th distrib.)} \qquad (44)$$

$$\overline{arb} = \frac{1}{H}\sum_{h=1}^{H}\left|rb_h\right| \qquad \text{(mean absolute relative bias)} \qquad (45)$$

where H=1000, the number of generating distributions, and K=30, the number of samples. An unbiased estimator should have $rb_h \cong 0$ for each *h*, obtaining a small value of $\overline{arb}$.

To be confident on the interpretation of the previous indices and, in particular, of $\overline{rse}$ as a global measure of performance, we computed also an "estimator's mean rank": for each sample we ranked the estimators on the strength of their ability to predict Err, the mean of these (30000) ranks is easily interpretable in order to compare the estimators.

10. Simulation experiments

We compared the estimators of Err by carrying out two simulations with different sample sizes: 120 and 500. To indicate a model, e.g. Regression Tree, applied to the sample size 500, we will use the synthetic notation TREE 500.
The list of estimators considered is the following:

- err, the resubstitution estimator;
- err^{RCV}, the Repeated 10-fold Cross Validation estimator, with 10 random start;
- err^{RCV*}, the same as err^{RCV} but with Burman's correction;
- err^{RHO}, the Repeated Hold-Out estimator, with 100 random splits;
- err^{LOO}, the Leave-one-out Cross Validation estimator;
- err^{B632}, the Bootstrap .632 estimator, with 100 bootstrap samples;
- err^{B632+}, the Bootstrap .632+ estimator, with 100 bootstrap samples;
- err^{PB}, the Parametric Bootstrap estimator, with 100 bootstrap samples;

All the previous estimators require similar computational times, except for err (very quick) and err^{LOO} (very slow for sample size = 500). Moreover err^{PB} also required a preliminary estimation step by nonparametric bootstrap (see Section 6).

It is useful to start from a general evaluation of the performance of the models on the two sample sizes. We can gather from table 10.1 that, for the smaller sample size, all models overfit. Indeed, *apparent error* is much lower than *extra-sample error*, this is true also for TREE 500.

Table 10.1
Extra-sample and *apparent error*. Mean on all generating distributions with respect to model and sample-size.

		Mean of *apparent error*	Mean of *extra-sample error*	Ratio
TREE	120	91.37	462.37	0,208
	500	125.97	340.14	0,384
PPR	120	141.33	395.22	0,358
	500	209.78	262.54	0,799
NEU	120	189,36	444,31	0,435
	500	121,53	151,98	0,810

The ratio $\overline{err}/\overline{Err}$ in table 10.1 can be interpreted as level of model overfitting: small values of the ratio correspond to high overfitting.

These results imply a significative level of overfitting, which is a quite common problem in data mining applications. It should be noted that presence of overfitting does not imply necessarely that we should use a simpler model, for example a tree with a smaller number of nodes.
To illustrate the problem, consider fig. 10.1. It shows the performance of a regression tree with sample size 120 with respect to the minimum splitting size of a node (splitmin). Larger the splitting size lesser the number of nodes. We have drawn one sample from 200 generating distributions with a strong signal (S/n= 4.5) and we have evaluated the performance of the regression tree for each value of splitmin.
The lowest values of *extra-sample error* has been obtained for the largest number of nodes, i.e. splitmin near the minimum value 2, but in this case the *apparent error* is near zero, to indicate presence of overfitting. This apparent incongruence is due to the high level of signal compared to the error.

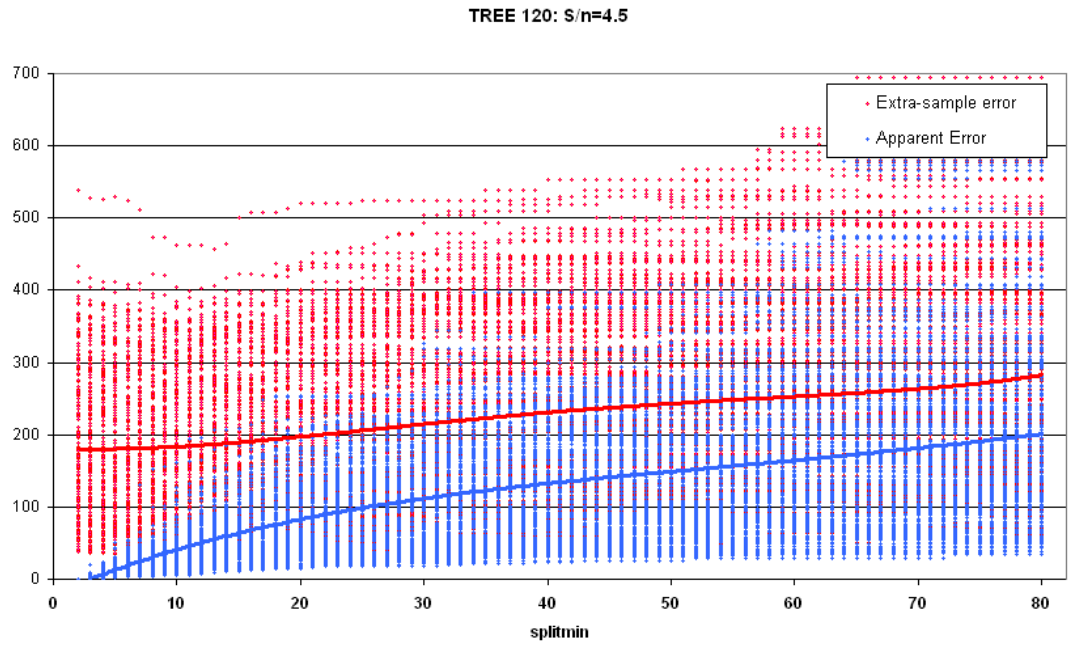

Fig. 10.1 – *Err* and *err* vs minimum splitting size, 200 generating distributions wit S/n ratio=4.5, for each splitting size. Regression Tree with sample size n=120.

From the table 10.1 we could also note that a greater sample-size reduces the difference between *extra-sample error* and *apparent error*. PPR and NEU with sample size = 500 do not show overfitting, in the other cases there is a significative level of overfitting.

Let us now consider the indices defined in the previous paragraph. In the following figures 10.2-10.10, we can see the values of rb_h (44) for RCV, RCV*, PB, B632,B632+, LOO, (computed as the mean of 30 samples for each of the 1000 generating distributions), by S/n ratio. Note the strong dependence of *PB, B632, B632+* on the S/n ratio. In the presence of severe overfitting and unstable models (TREE 120), we can see the remarkable improvement on B632 obtained by B632+ for small values of the S/n ratio. However, for the other models, B632+ were worse than B632. Conversely, we did not observe any remarkable influence of the level of linearity (measured by R^2) with respect to the estimator performance (see fig. 10.11 and 10.12).

Considering the more stable model PPR with the larger sample, we obtain fig. 10.9 and 10.10. Note the good performance of RCV*, PB and LOO with respect to the biased B632.

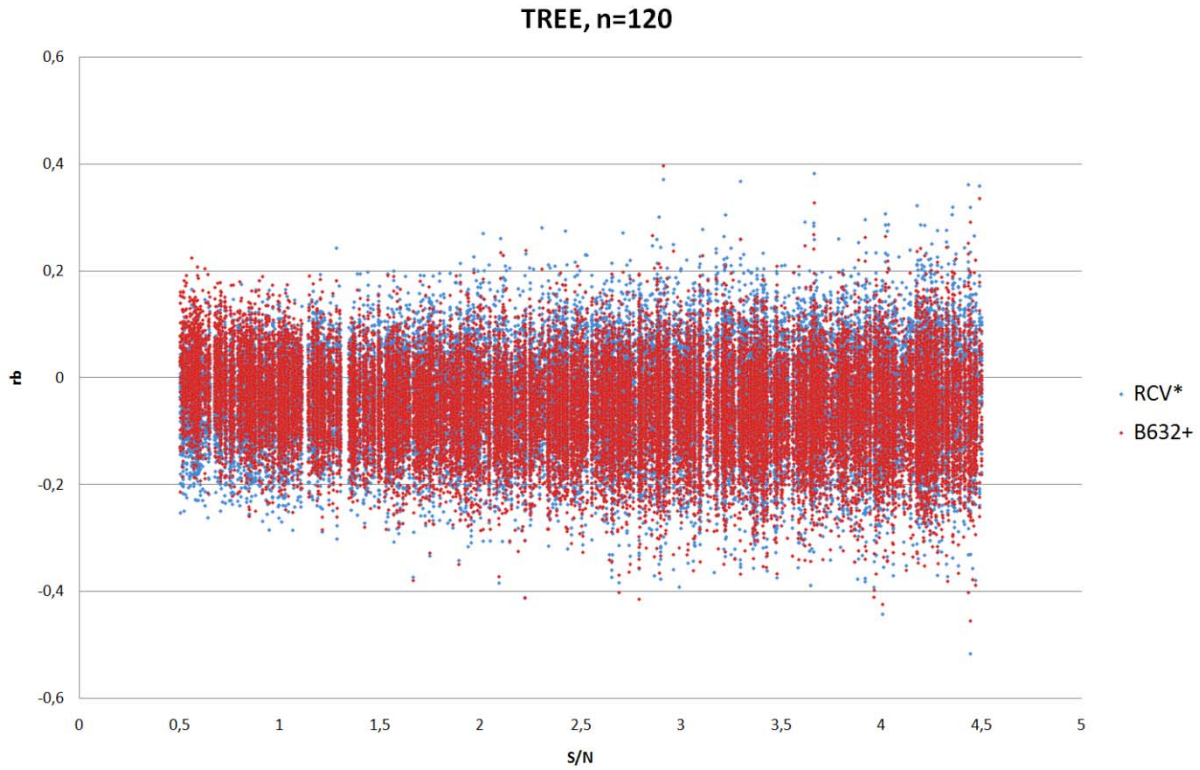

Fig. 10.2 – $rb_{(h,k)}$ vs S/n ratio of RCV* and B632+ for 30.000 samples, 1000 generating distributions. TREE with sample size n=120.

In this plot we can see the sample value components of rb_h (44) for B632 and RCV, for each of 30.000 samples. The predictive model is the Regression Tree model (for details see section 9). Given the S/n ratio, we can see that RCV* has small bias but large variability.*

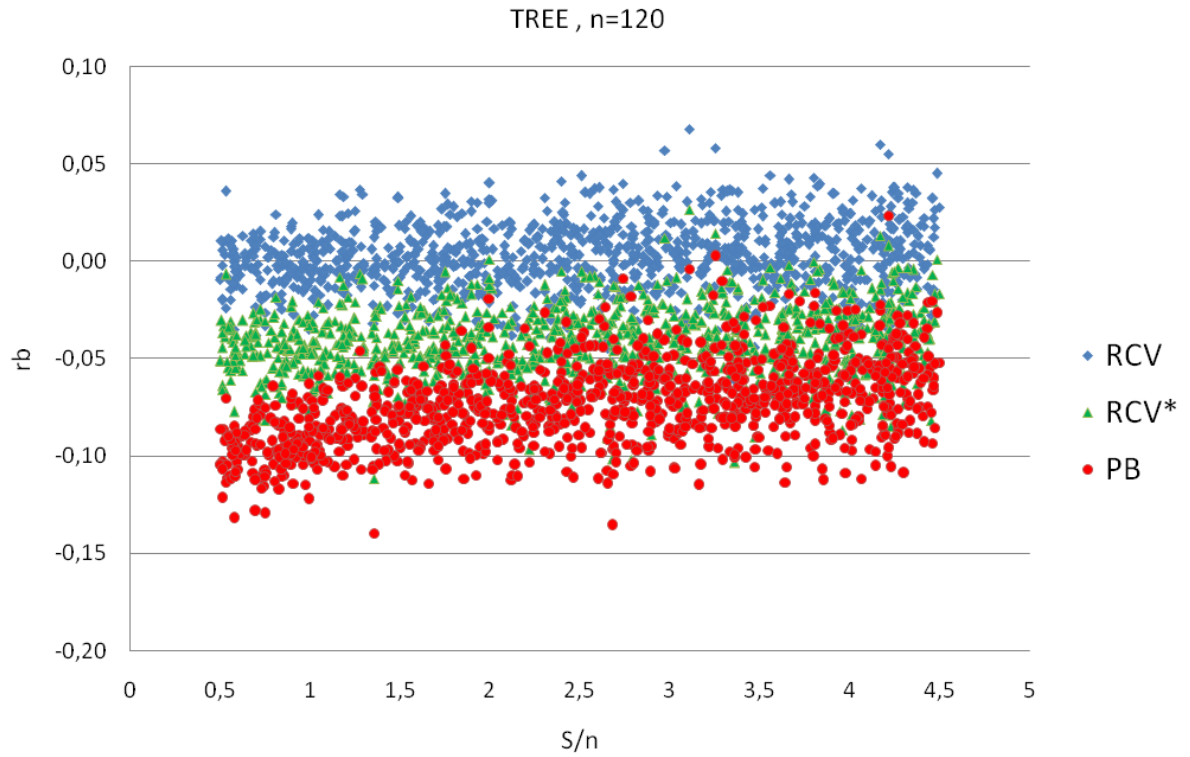

Fig. 10.3 – *rb* vs S/n ratio. TREE with sample size n=120

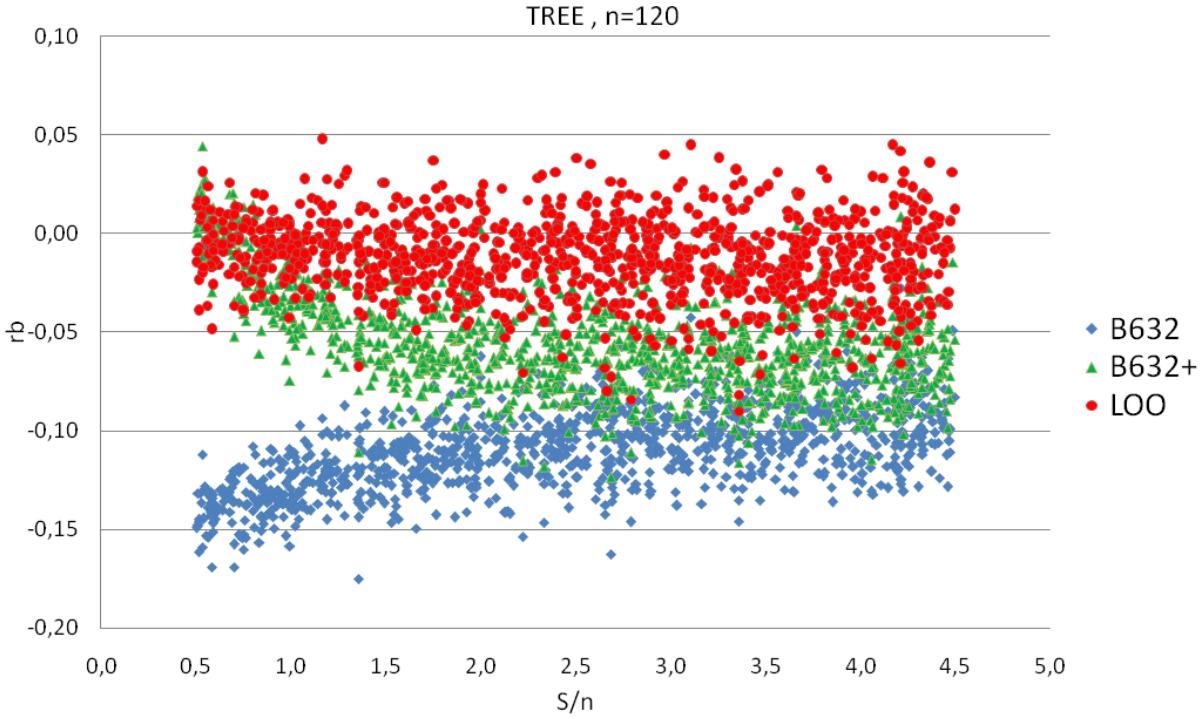

Fig. 10.4 – *rb* vs S/n ratio. TREE with sample size n=120

In these plots we can see the value of rb (44) *for RCV, RCV*, PB, B632,B632+, LOO, computed as mean of 30 samples for each of 1000 generating distributions. The predictive model is the Tree Regression model with sample size=120 (for details see section 9). PB, B632, B632+ show a strong dependence by S/n ratio.*

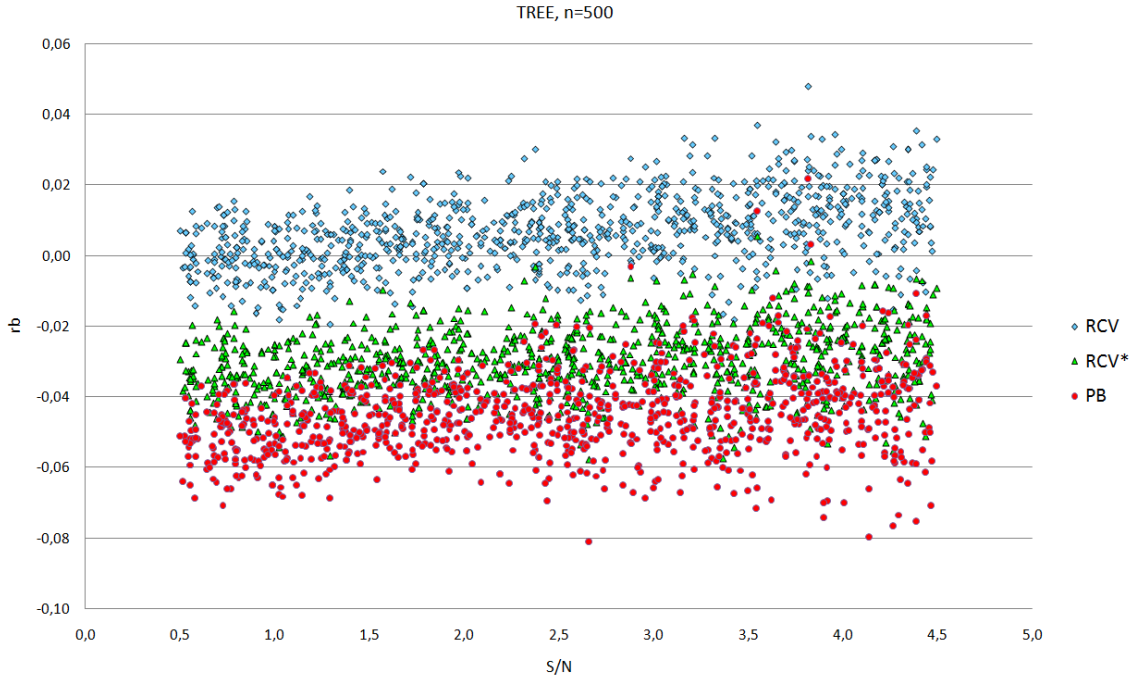

Fig. 10.5 – *rb* vs S/n ratio. TREE with sample size n=500

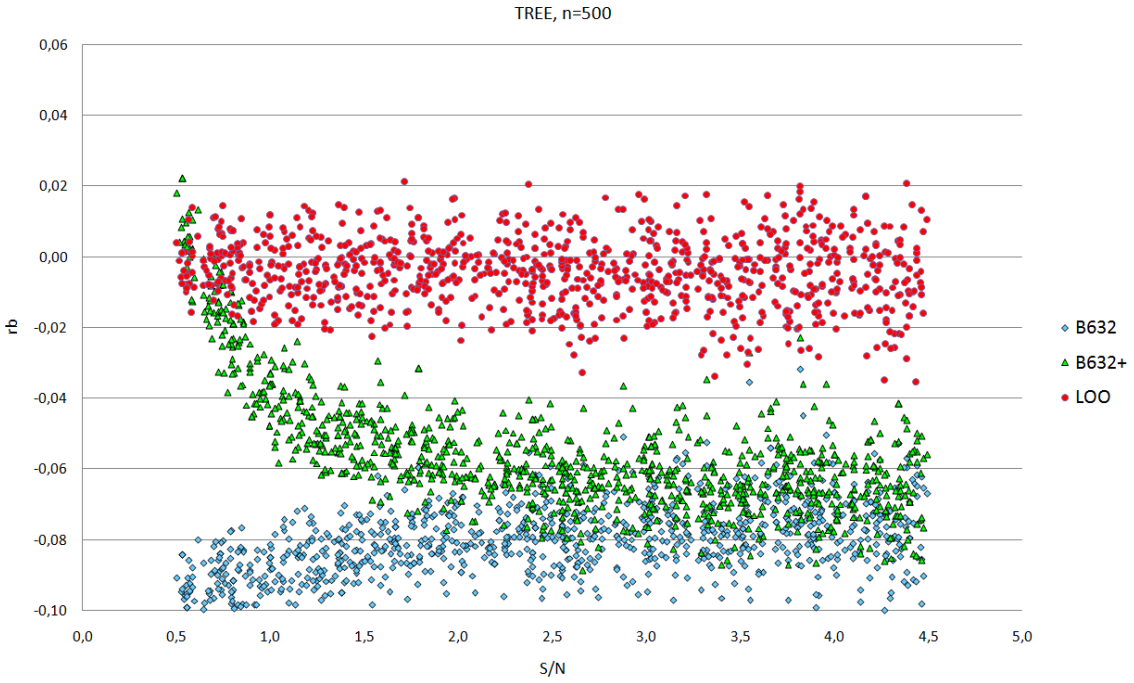

Fig. 10.6 – *rb* vs S/n ratio. TREE with sample size n=500

In these plots we can see the value of rb for RCV, RCV, PB, B632,B632+, LOO computed as mean of 30 samples for each of 1000 generating distributions. The predictive model is the Tree Regression model (for details see section 9). B632,B632+ show a strong dependence by S/n ratio.*

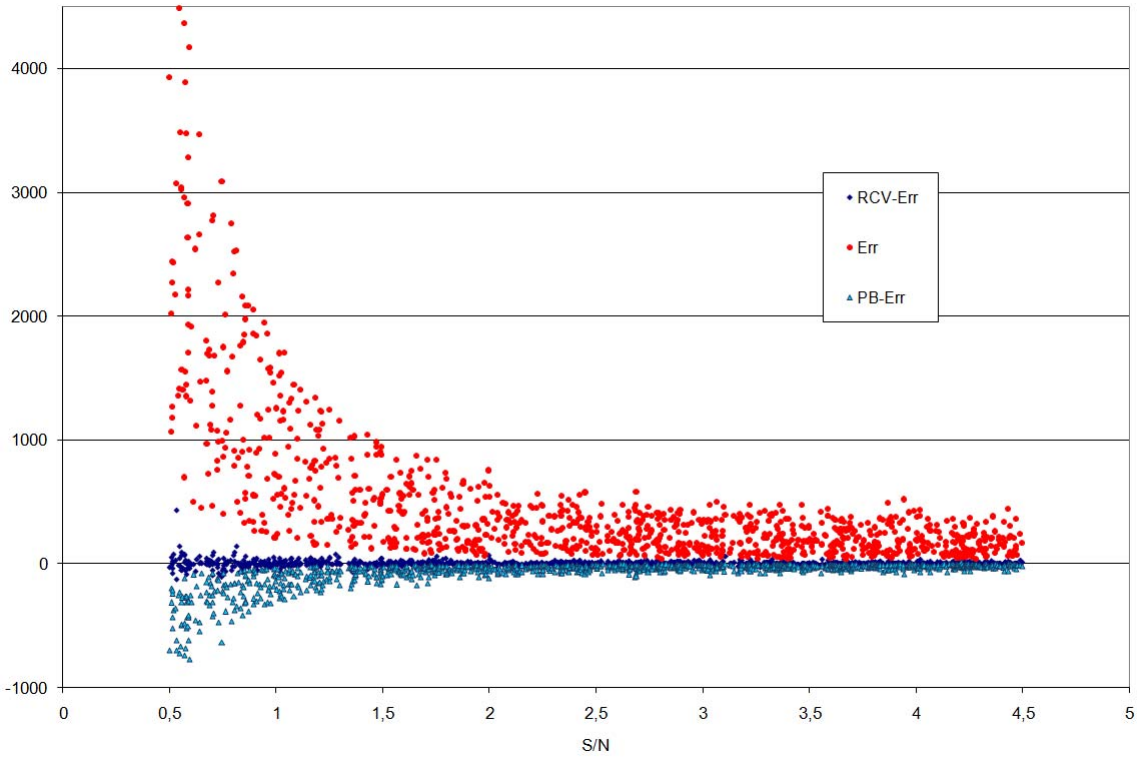

Fig. 10.7 – It is represented the difference between RCV and PB estimate with *Extra Sample Error*. We reported also the value of *Extra Sample Error* (red points). The model used is Regression Tree with sample size=120.

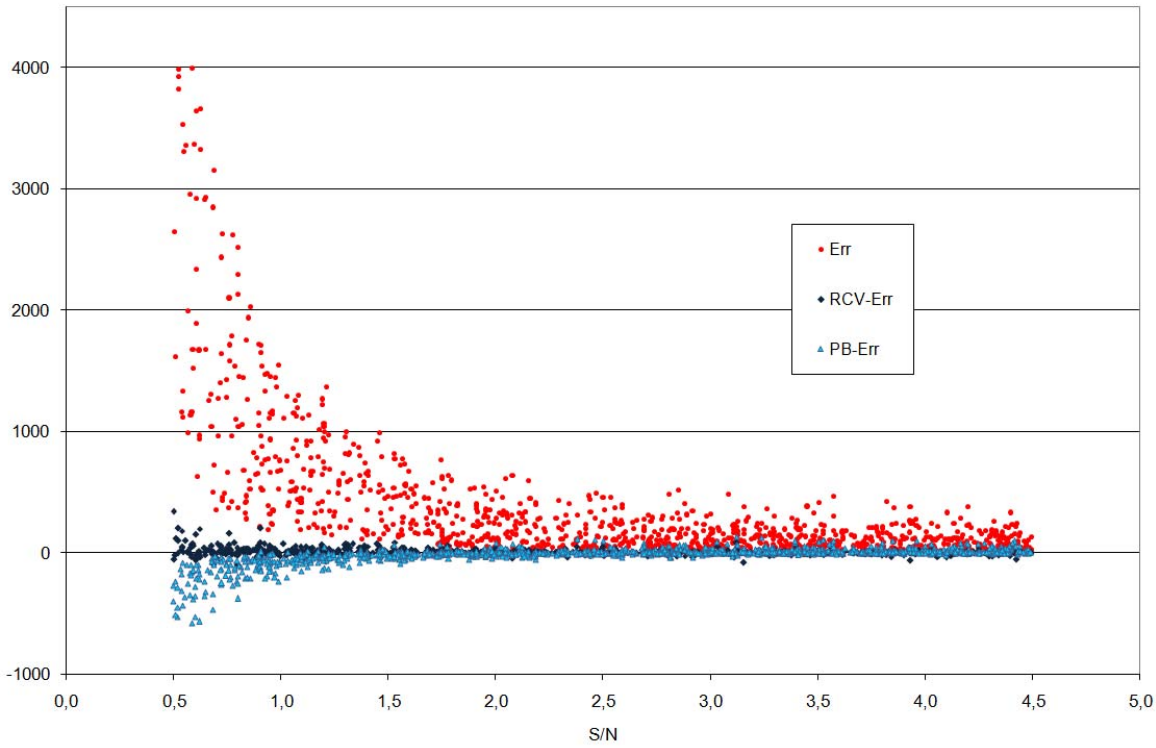

Fig. 10.8 – It is represented the difference between RCV and PB estimate with *Extra Sample Error*. We reported also the value of *Extra Sample Error* (red points). The model used is Projection Pursuit Regression with sample size=120.

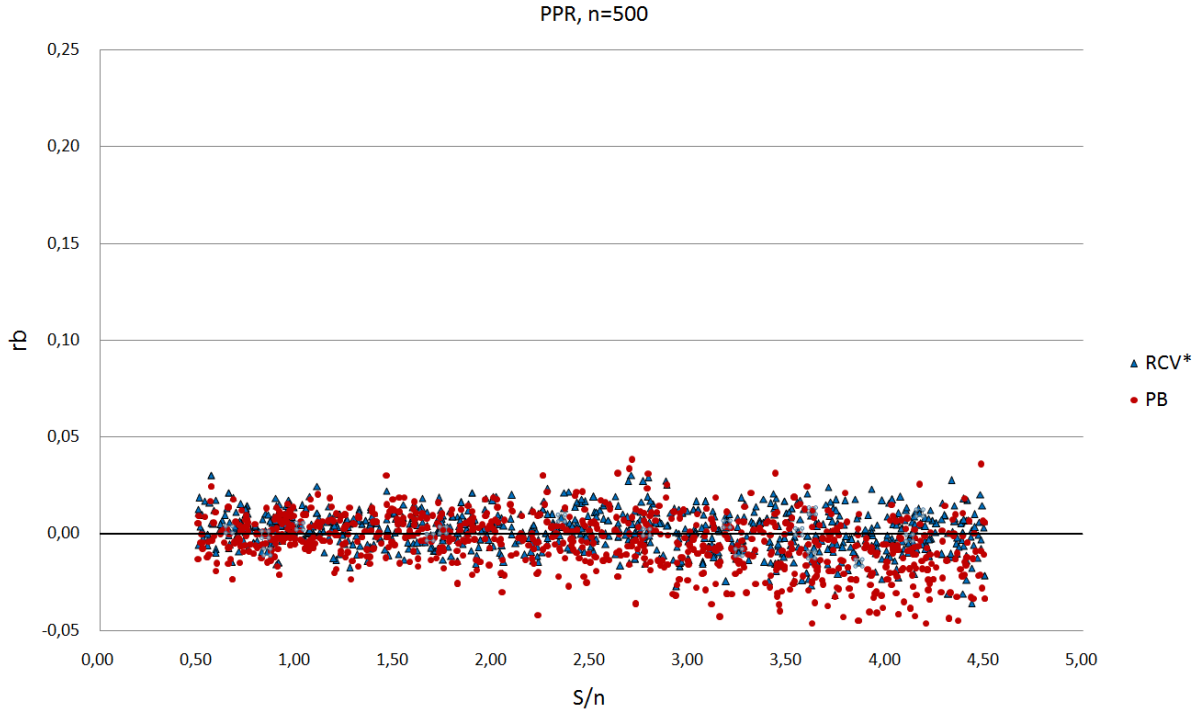

Fig. 10.9 – rb_h vs S/n ratio. PPR with sample size n=500

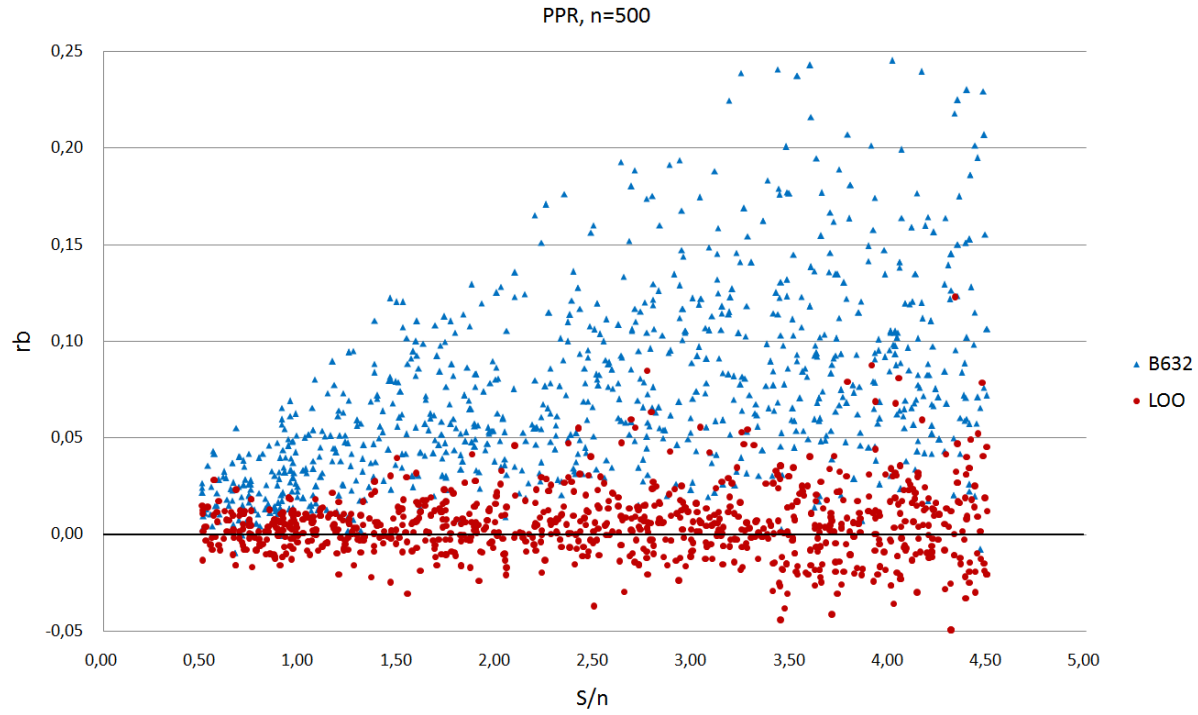

Fig. 10.10 – rb_h vs S/n ratio. PPR with sample size n=500

In table 10.2 and 10.3 we show for each estimator the value of $\overline{arb}$, measuring the absolute relative bias, and the value of $\overline{rse}$, which is an overall bias-variance relative index. A good estimator should have $\overline{arb}$ and $\overline{rse}$ as small as possible. The results are reported for both simulations, the first with sample-size 120 and the second with sample-size 500.

Table 10.2.
Mean absolute relative bias $\overline{arb}$. Mean on all generating distributions with respect to estimator, model and sample-size.

$\overline{arb}$	TREE		PPR		NEU	
	sample size		*sample size*		*sample size*	
	120	500	120	500	120	500
RCV	**0.014**	**0.010**	0.030	0.019	0.079	0.023
RCV*	0.043	0.030	**0.022**	**0.008**	**0.036**	**0.010**
PB	0.074	0.045	0.062	0.012	0.057	**0.010**
B632	0.109	0.080	0.124	0.074	0.057	0.021
B632+	0.056	0.055	0.182	0.082	0.080	0.022
LOO	**0.018**	**0.008**	**0.023**	0.013	--------	--------
RHO	0.037	0.034	0.122	0.045	0.210	0.071

Table 10.3.
Mean relative root squared error $\overline{rse}$. Mean on all generating distributions with respect to estimator, model and sample-size.

$\overline{rse}$	TREE		PPR		NEU	
	sample size		*sample size*		*sample size*	
	120	500	120	500	120	500
RCV	**1.658**	**1.828**	1.181	1.315	1.271	1.271
RCV*	1.733	2.149	**1.091**	**0.962**	**0.937**	0.829
PB	1.956	2.485	1.183	**0.945**	**0.948**	**0.780**
B632	2.286	3.486	1.563	1.773	1.051	1.054
B632+	1.732	2.669	2.185	1.961	1.206	1.079
LOO	1.856	2.038	1.260	1.323	--------	--------
RHO	1.807	2.341	1.674	1.562	2.182	1.947

Table 10.4.
The estimators' comparative performance: estimator mean ranks. Lower values indicate better estimators.

	TREE		PPR		NEU	
	sample size		*sample size*		*sample size*	
	120	500	120	500	120	500
RCV	**2.87**	**2.63**	3.21	3.54	3.73	3.93
RCV*	3.16	**2.87**	**2.89**	**2.74**	**2.63**	2.64
PB	3.67	3.40	3.30	**2.75**	**2.71**	**2.45**
B632	4.88	5.42	3.99	4.52	3.09	3.16
B632+	3.13	3.86	5.14	5.39	3.68	3.60
LOO	3.30	**2.82**	3.35	3.50	------	------
AE	6.99	7.00	6.11	5.57	5.17	5.22

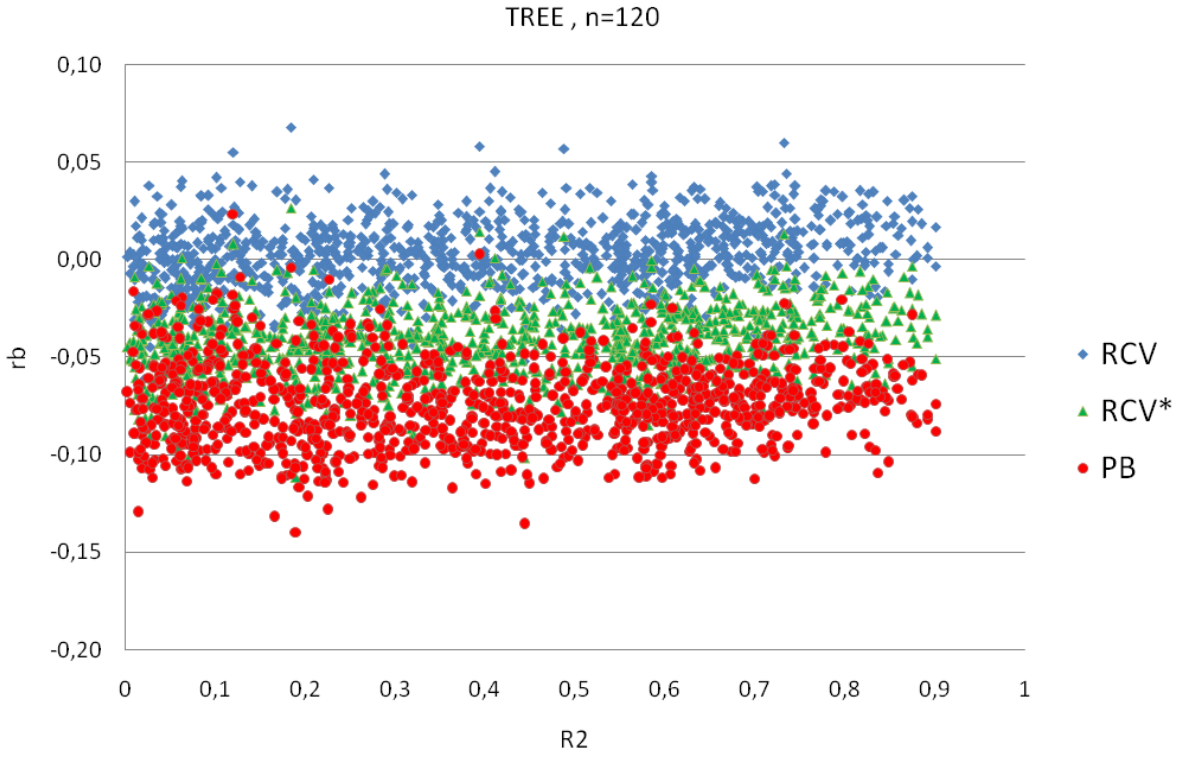

Fig. 10.11 – *rb* vs R^2 . TREE with sample size n=120

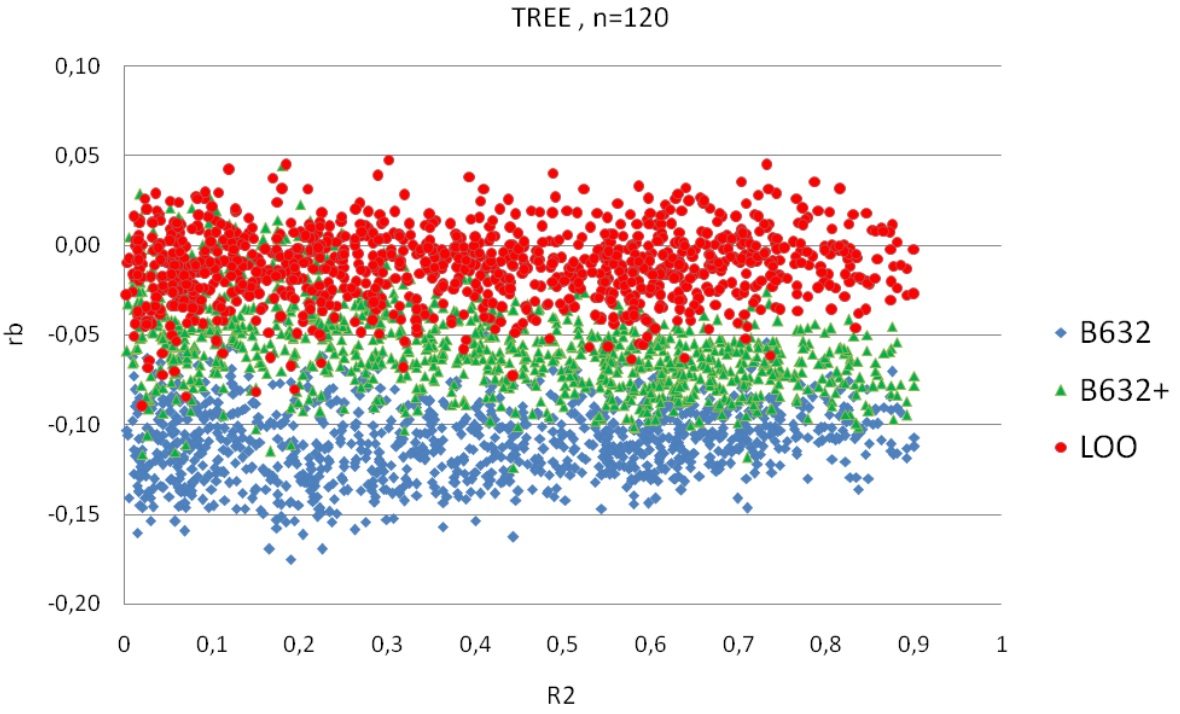

Fig. 10.12 – *rb* vs R^2 . TREE with sample size n=120

In these plots we can see the value of rb_h (44) for RCV, RCV, PB,B632, B632+, LOO, computed as mean of 30 samples for each of 1000 generating distributions. The predictive model is the Tree Regression model (for details see section 9).*

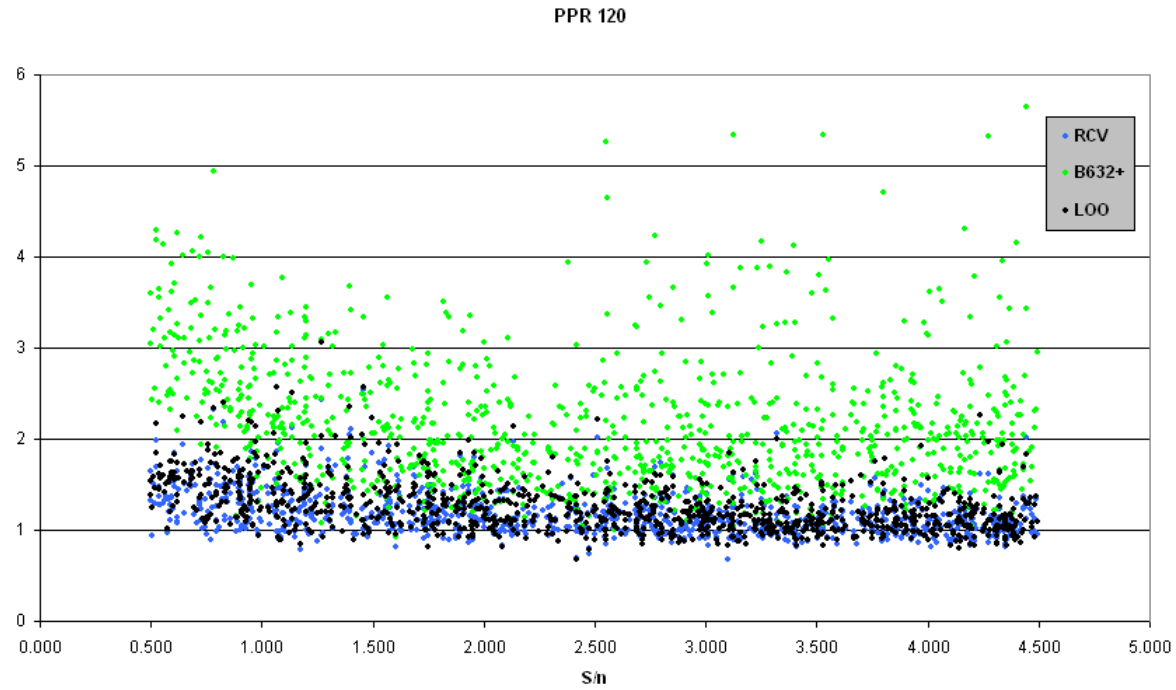

Fig. 10.13 – rse_h vs S/n ratio, RCV, B632+, LOO. PPR with sample size n=120

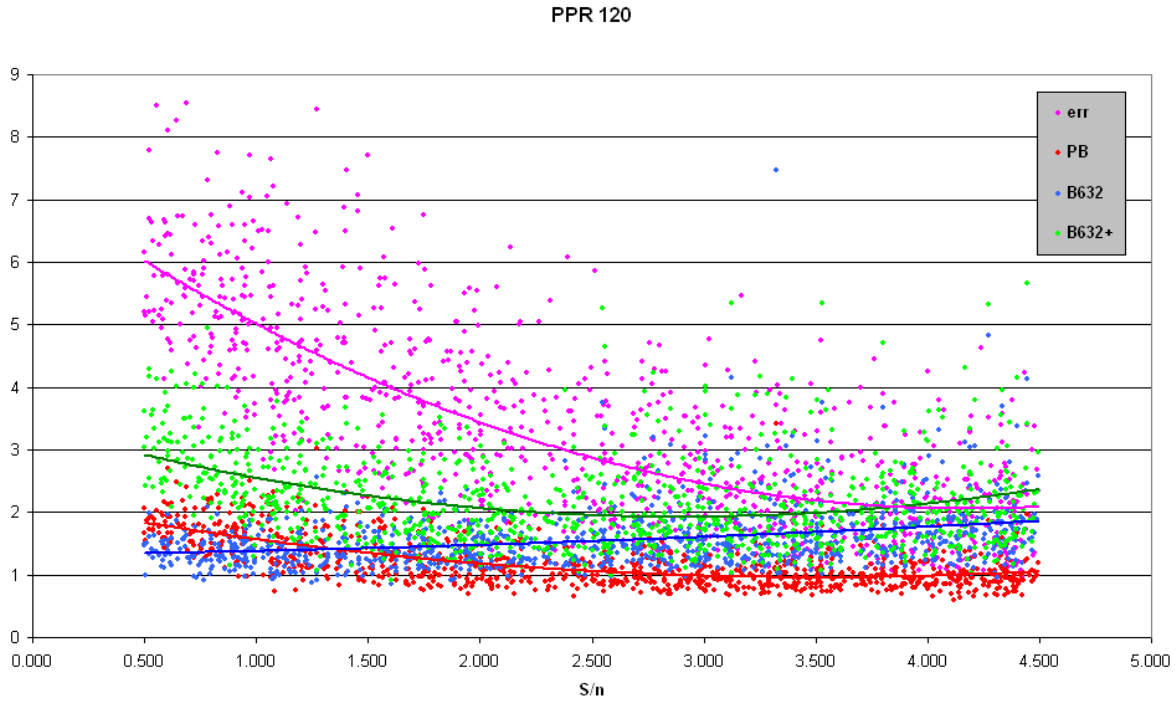

Fig. 10.14 – rse_h vs S/n ratio, *err*, PB, B632, B632+. PPR with n=120

In the following tables we compared the variability of the different estimators. In particular we considered the mean of the standard deviation of each estimator.

Table 10.5.
Comparison of the estimator variability. Sample size = 120.

	Err	RCV	err	RCV*	PB	B632	B632+	LOO
TREE	45.8098	74.1814	21.4261	69.0100	57.8333	50.6303	66.1099	84.8805
PPR	75.1120	62.0773	36.5505	66.2129	59.9678	62.0658	82.1610	73.3496
NEU	111.6404	73.5409	59.1412	86.7476	75.8818	54.4213	68.3934	NA

Table 10.6.
Comparison of the estimator variability. Sample size = 500.

	Err	RCV	err	RCV*	PB	B632	B632+	LOO
TREE	15.4128	25.9033	12.1516	24.9253	22.6055	19.2520	22.8302	30.3857
PPR	23.6408	19.6600	23.8004	27.0591	26.3019	20.9158	21.9575	21.9735
NEU	14.0461	8.7822	10.5144	12.1127	11.5127	8.4300	8.5865	NA

11. Extimator's behaviour for ill-specified models

What happen when we have more complex data or the model is ill-specified?
Consider the following case (inspired from Friedman, 1991):

$$y = 10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right) + 5x_4 + \varepsilon \tag{46}$$

where the covariates were randomly generated from a uniform distribution.
We want ε to be dependent from $f(\mathbf{x})$, so we introduced a "non-observable" random variable $\tau : \tau = \chi^2_{(4)} \cdot f(\mathbf{x}) \cdot 0.2$, then, given the random s/n ratio and σ_ε as described in the section 4, we set

$$\varepsilon = \tau \frac{\sigma_\varepsilon}{\sigma_\tau} \tag{47}$$

As an example, we show the bias introduced by ε in the following plot (a sample of 1000 cases for one generating distribution with S/n=2.0).

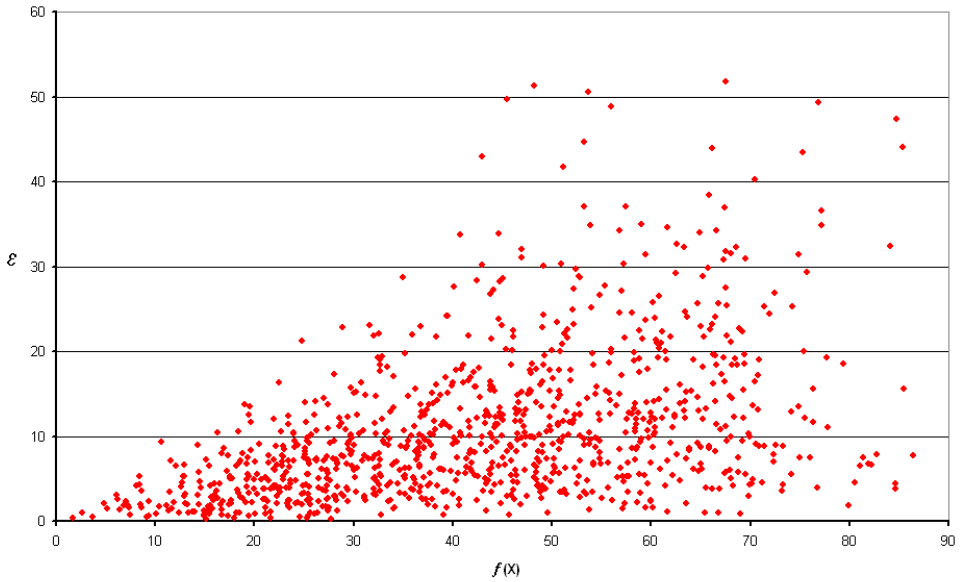

Fig. 11.1 – *Noise* vs Signal, one sample (1000 cases) from one generating distribution

Finally we have obtained the results showed in the following tables. Note that the number of generating distributions considered is not high (300), but this is a useful test to confirm and consider more reliable the previous results.

In table 11.1 and 11.2 they are reported the results which can be compared with the values shown in tables 10.2-10.3.

Table 11.1.
Mean absolute relative bias $\overline{arb}$. Mean on all generating distributions with respect to estimator, model and sample-size.

$\overline{arb}$	TREE		PPR	
	sample size		*sample size*	
	120	500	120	500
RCV	**0.015**	0.013	0.030	0.026
RCV*	0.033	0.025	**0.019**	**0.010**
PB	0.074	0.052	0.035	0.033
B632	0.104	0.080	0.106	0.102
B632+	0.059	0.061	0.147	0.107
LOO	**0.013**	**0.007**	0.024	0.017
RHO	0.047	0.042	0.125	0.053

Table 11.2.
Mean relative root squared error $\overline{rse}$. Mean on all generating distributions with respect to estimator, model and sample-size.

$\overline{rse}$	TREE		PPR	
	sample size		*sample size*	
	120	500	120	500
RCV	**1.588**	**1.542**	1.090	1.049
RCV*	1.646	1.748	**0.979**	**0.485**
PB	1.980	2.375	1.028	0.600
B632	2.320	3.130	1.381	1.412
B632+	1.759	2.561	1.766	1.489
LOO	1.830	1.777	1.170	1.034
RHO	1.810	2.285	1.556	1.189

12. Conclusions

The simulations were based on samples drawn from 1000 generating distributions with different characteristics, considering different levels of signal-to-noise ratio, overfitting, and non-linearity of data. The comparison between the estimated and the true *prediction error*, calculated on such a large number of samples, allows us to investigate the performance of the estimators better than, for example, in Kohavi (1995). The choice of three models with different levels of fitting casts new light upon estimator performance, as it is know that the stability of the model influences significantly the performance of the estimators (Elisseeff et al., 2003). From this point of view Projection Pursuit Regression, for a small number of terms, and Neural Networks, with few hidden nodes, are stable methods while Regression Trees, without pruning, is unstable. Moreover the use of two different sample sizes allows to evaluate their effect on the *prediction error* estimators.

With these simulations, we have studied the effect of the S/n ratio and level of overfitting on the performance of the estimators, which had not previously been considered in the literature. The most evident result is that the repeated corrected 10-fold cross-validation RCV* proposed by Burman (1989) outperformed the other estimators overall. The only estimator which appears to be competitive is the Parametric Bootstrap (Efron, 2004), especially with large samples and stable models.

We should also note the poor performance of the estimators based on non-parametric bootstrap B632 and B632+, or on repeated Hold-Out, RHO, and the unsatisfactory results obtained by the leave-one-out. Despite the substantial computational resources required, LOO appears to be quite unbiased but is never the most convenient estimator to use essentially due to its high variability. It could be observed that B632 and B632+ would have a better performance with a larger number of bootstrap sub-samples, but this would be true also for the other estimators RCV, RCV*, PB, RHO. Moreover B632+ outperforms B632 only in case of severe overfitting.

The positive performance of RCV on Regression Trees is due, paradoxically, to a known defect of k-fold Cross-Validation which tends to overestimate the *extra-sample error*. Since Regression Trees highly overfit our data and all estimators tend to underestimate *Err*, this behaviour makes RCV, in this case, the least biased estimator.

In the same way, we have also analysed non-Normal heteroskedastic data in section 11, relaxing the hypothesis $\varepsilon \sim N\left(0,\sigma_\varepsilon^2\right)$ we made in Section 10. As expected, the main result is the downgrade of the performance of Parametric Bootstrap which requires this hypothesis in order to estimate the bootstrap density.

Summarizing, from the two simulations, we obtained the following results:

- Level of overfitting and signal/noise ratio influence the performance of some estimators, while the level of linearity is not relevant.
- The best performance was obtained by RCV*.
- Leave-one out is not always the less biased estimator and it has relevant variability and heavy computational time.
- Resampled estimators RCV, RCV*, RHO have lower variability than the simple estimators CV, CV*, HO
- Stability of the model and sample size influence the results
- Non-parametric B632 and B632+ have complessively a poor performance, Parametric Bootstrap is undoubtedly a better estimator (in particular with large sample and stable models).

References

Akaike, H., 1973. Information Theory and an extension of the Maximum Likelihood Principle, Second Intern. Symposium on Information Theory, 267-281.

Bengio, Y., Grandvalet, Y., 2003. No unbiased estimator of the variance of K-fold cross-validation, Journal of Machine Learning Research, v. 5, 1089-1105.

Borra, S., Di Ciaccio, A., 2004. Methods to Compare Nonparametric Classifiers and to Select the Predictors, in New Developments in Classification and Data Analysis, M. Vichi, et al. editors, Springer-Verlag, Berlin, 11-20.

Bousquet, O., Elisseeff, A., 2002. Stability and generalization, Journal of Machine Learning Research, 2, 499-526.

Breiman, L., 1992. The little bootstrap and other methods for dimensionality selection in regression: x-fixed prediction error. Journal of the American Statistical Association, 87 (419), 738-754.

Breiman, L., 1996. Heuristic of instability and stabilization in model selection, The Annals of Statistics, v. 24, n. 6, 2350-2383.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., 1984. Classification and Regression Trees, Monterey: Wadsworth and Brooks-Cole, New York.

Breiman, L., Spector, P., 1992. Submodel selection and evaluation in regression: The X-random case, International Statistical Review, 60, 291-319.

Burman, P., 1989. A comparative study of ordinary cross-validation, v-fold cross-validation and repeated learning-testing methods, Biometrika, 76, 3, 503-514.

Burman, P., 1990. Estimation of optimal transformations using v-fold cross validation and repeated learning-testing methods, Sankhya, series A, v. 52, 314-345.

Burman, P., 1996. Model fitting via testing, Statistica Sinica 6, 589-601.

Daudin, J.J., Mary-Huard, T., 2008. Estimation of the conditional risk in classification: The swapping method, Computational Statistics and Data Analysis, 52, 3220-3232.

Davison, A.C., Hinkley, D.V., 1997. Bootstrap Methods and their Applications, Cambridge University Press, Cambridge.

Devroye, L., Györfi, L., Lugosi, G., 1996. A Probabilistic Theory of Pattern Recognition, New York/Berlin: Springer-Verlag.

Devroye, L., Wagner, T.J., 1979. Distribution-free performance bounds with the resubstitution error estimate, IEEE Transactions on Information Theory, vol. IT-25 n.2.

Dietterich, T. G., 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, Neural Computation, 10 (7) 1895-1924.

Dudoit, S., van der Laan, M., 2005. Asymptotics of Cross-Valdated Risk Estimation in Estimator Selection and Performance Assessment, Statistical Methodology, 2(2), 131-154.

Efron, B., 1983. Estimating the error rate of a prediction rule: Improvement on cross-validation, Journal of the American Statistical Association, 78, 316-331.

Efron, B., 1986. How biased is the *apparent error* Rate of a Prediction Rule?, Journal of the American Statistical Association, 81, 461-470.

Efron, B., 2004. The Estimation of Prediction Error: Covariance Penalties and Cross-Validation, Journal of the American Statistical Association, v. 99, n. 467, 619-632.

Efron, B., Tibshirani, R.J., 1993. An Introduction to the Bootstrap, London: Chapman & Hall.

Efron, B., Tibshirani, R., 1997. Improvements on Cross-Validation: The .632+ Bootstrap Method, Journal of the American Statistical Association, Vol. 92, n. 438, 1997.

Elisseeff, A., Pontil, M., 2003. Leave-one-out error and stability of learning algorithms with applications, in Advanced in Learning Theory: Methods, Models and Applications, NATO Science Series III: Computer and Systems Sciences, Vol. 190, Suykens et. al. eds., IOS Press.

Friedman, J.H., 1985. SMART User's Guide. Dept. of Statistics, Technical Report LCS 1, Stanford University.

Friedman, J. H, 1991. Multivariate Adaptive Regression Splines, Annals of Statistics 19, 1-67.

Friedman, J.H., Stuetzle, W., 1981. Projection Pursuit Regression, JASA, 76, 817-823.

Gascuel, O., Caraux, G., 1992. Distribution-free performance bounds with the resubstitution error estimate, Pattern Recognition Letters, North-Holland, 13, 757-764.

Hastie, T., Tibshirani, R., 1990. Generalized Linear Models, Chapman Hall, London.
Hastie, T., Tibshirani, R., Friedman, J.H., 2001. The elements of Statistical learning: Data Mining, Inference and Prediction. Springer-Verlag, 2001.
Kearns, M., 1997. A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for the Training-Test Split, Neural Computation, 9(5), 1143-1161.
Kearns, M., Ron, D., 1999. Algorithmic Stability and Sanity-Check Bounds for LeaveOne -Out Cross-Validation, Neural Computation, 11(6):1427-1453.
Kohavi, R., 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, International Joint Conference on Artificial Intelligence, 1995.
Larsen, J., Goutte, C., 1999. On optimal data split for generalization estimation and model selection, in Neural Networks for Signal Processing IX. Proceedings of the 1999 IEEE Signal Processing Society Workshop:225 – 234.
Mallows, C., 1973. Some comments on Cp, Technometrics, 15, 661-675.
Molinaro, A.M., Simon, R., Pfeiffer, R.M., 2005. Prediction Error Estimation: A Comparison of Resampling Methods, Bioinformatics, Vol.21, n.15.
Rogers, W. H., Wagner, T. J., 1978. A fine sample distribution-free performance bound for local discrimination rules. The Annals of Statistics, 6(3):506-514.
Schwarz, G., 1978. Estimating the dimension of a Model, The Annals of Statistics, 6, 461-464.
Shao, J., 1993. Linear Model Selection by Cross Validation, Journal of the American Statistical Association, Vol. 88, n. 422.
Shao, J., 1997. An asymptotic theory for linear model selection, Statistica Sinica, 7, 221-264.
Shen, X., Ye, J., 2002. Adaptive Model Selection, Journal of the American Statistical Association, 97, 210-221.
Stein, C., 1981. Estimation of the Mean of a Multivariate Normal Distribution. The Annals of Statistics, 9, 1135–1151.
Stone, M., 1977. Asymptotics for and against cross-validation, Biometrika, 64, 1, 29-35.
Tibshirani, R., Knight, K., 1999. Model search and inference by bootstrap bumping, J. Comp. and Graph. Stat. 8, 671-686.
van der Laan, M.J., Dudoit, S., 2003. Unified Cross-Validation Methodology For Selection Among Estimators and a General Cross-Validated Adaptive Epsilon-Net Estimator: Finite Sample Oracle Inequalities and Examples, U.C. Berkeley Division of Biostatistics, Working Paper Series n. 130.
Vapnik, V., 1998. Statistical Learning Theory, Wiley, New York.
Ye, J., 1998. On Measuring and Correcting the Effects of Data Mining and Model Selection, Journal of the American Statistical Association, 93, 120-131.
Zhang, B.P., 1993. Model selection via Multifold Cross Validation, The Annals of Statistics, Vol.21, n.1, 299-313.
Zhang, C., 2003. Calibrating the degrees of freedom for automatic data smoothing and effective curve checking. Journal of the American Statistical Association, v. 98, n. 463, 609-628.
Zhang, C., 2008. Assessing prediction error of nonparametric regression and classification under Bregman divergence, Scandinavian Journal of Statistics, V. 35 Issue 3, 496 - 523.
Zhang, T., 2001. A leave-one-out cross validation bound for kernel methods with applications in learning. In COLT 01, 427-443.