EE 649 Pattern Recognition

Error Estimation

Ulisses Braga-Neto

ECE Department

Texas A&M University

Main Ideas

- How does one estimate the classification error of a given designed classifier?
- Is a good classifier good if we have no means of accurately estimating its classification error?
- If we knew the true distribution of the data, then we can in principle compute the error of a classifier exactly.
- But we generally do not know that and are given only training data, or (rarely) testing data.
- Error estimation is involved in classifier design itself (implicitly) and in feature selection.

Classification Errors (Review)

- Given any classifier $\psi : R^d \to \{0,1\}$, its error is:

$$\epsilon\left[\psi\right] = P\left(Y \neq \psi(X)\right) = E\left[|Y - \psi(X)|\right]$$

- Bayes error (minimum classification error):

$$\epsilon^* = \epsilon\left[\psi^*\right] = E\left[|Y - \psi^*(X)|\right]$$

- Designed classifier error for classification rule Ψ_n:

$$\epsilon_n = \epsilon\left[\psi_n\right] = E\left[|Y - \psi_n(X)|\right] = E\left[|Y - \Psi_n(X; S_n)| \,\middle|\, S_n\right]$$

- Expected classification error:

$$\mu_n = E\left[\epsilon_n\right] = E\left[E\left[|Y - \Psi_n(X; S_n)| \,\middle|\, S_n\right]\right]$$

Some Observations

- The Bayes error ϵ^* provides a universal bound on classification performance, but it is usually very difficult to estimate with any accuracy.
- The designed classifier error ϵ_n is the most important one for practical purposes; this is usually *the error we would like to estimate.*
- The expected classification error $E[\epsilon_n]$ is of limited practical use; it is used to compare the performance of classification rules or to answer theoretical questions about a classification rule (such as consistency).
- However, the expected error can become important in practice if $\epsilon_n \approx E[\epsilon_n]$, that is, if the *error variance* $\text{Var}(\epsilon_n)$ is very small.

Error Estimation Rule

- An *error estimation rule* is a mapping $\Xi_n : (\Psi_n, S_n, \xi) \mapsto \hat{\varepsilon}_n$, where
  - Ψ_n is a given classification rule;
  - S_n is sample data;
  - ξ are *internal random factors*;
  - $0 \leq \hat{\varepsilon}_n \leq 1$ is an *error estimator*.
- A *pattern recognition rule* (Ψ_n, Ξ_n) consists of a classification rule Ψ_n and an error estimation rule Ξ_n.
- A *pattern recognition model* $(\psi_n, \hat{\varepsilon}_n)$ is a realization of a pattern recognition rule given data.

Error Estimation Rule - II

- The error estimator $\hat{\epsilon}_n$ is a random variable, through S_n and ξ. The *error estimate* is the value of $\hat{\epsilon}_n$ given realizations of S_n and ξ and is thus a real number.
- Unless otherwise stated, $\hat{\epsilon}_n$ is meant to be an approximation to the designed classifier error $\epsilon_n = E\left[|Y - \psi_n(X)|\right]$.
- An error estimator can be:
  - *Non-randomized*: Given the training data S_n, the estimator $\hat{\epsilon}_n$ is fixed (there are no internal random factors ξ).
  - Randomized: Given the training data S_n, the estimator $\hat{\epsilon}_n$ is not a fixed quantity. It is still a random variable through ξ.

Variance of Error Estimators

The *internal variance* V_{int} of $\hat{\epsilon}_n$ measures the variability due only to the internal random factors.

$$V_{\text{int}} = \text{Var}(\hat{\epsilon}_n | S_n)$$

For non-randomized $\hat{\epsilon}_n$, we have $V_{\text{int}} = 0$. Randomized error estimators however have this extra source of variability.

The *full variance* $\text{Var}(\hat{\epsilon}_n)$ of $\hat{\epsilon}_n$ measures the variability due to both the training data S_n (which depends on the data distribution and implicitly also on Ψ_n) and the internal random factors ξ.

Variance of Error Estimators - II

Using the conditional variance formula

$$\text{Var}(X) = E[\text{Var}(X|Y)] + \text{Var}(E[X|Y])$$

with $X = \hat{\epsilon}_n$ and $Y = S_n$, one gets:

$$\text{Var}(\hat{\epsilon}_n) = E[V_{\text{int}}] + \text{Var}(E[\hat{\epsilon}_n|S_n])$$

The second term on the right-hand side is the one that includes the variability due to the random training data S_n.

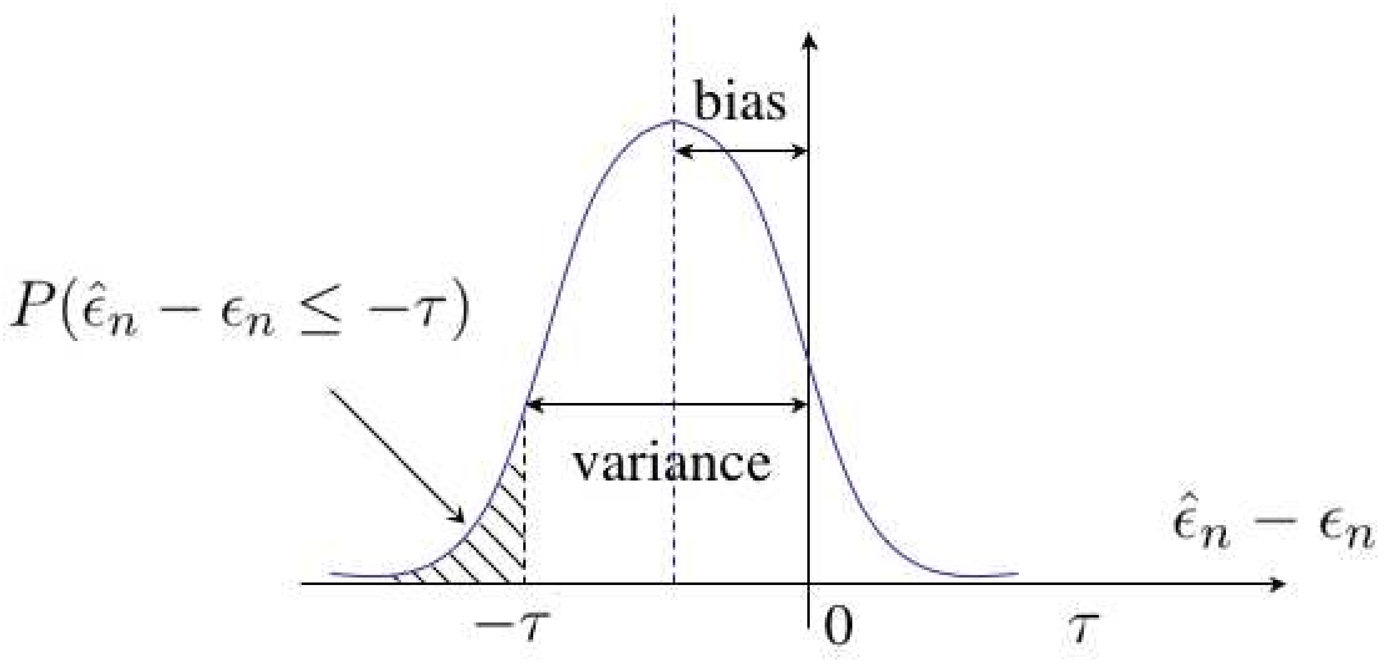
For non-randomized $\hat{\epsilon}_n$, we have $V_{\text{int}} = 0$ and $E[\hat{\epsilon}_n|S_n] = \hat{\epsilon}_n$.

For randomized $\hat{\epsilon}_n$, the first term on the right-hand side has to be made small. This is usually done through intensive computation, a characteristic drawback of such estimators.

Relationship Between $\hat{\epsilon}_n$ and ϵ_n

The relationship between $\hat{\epsilon}_n$ and ϵ_n can be completely specified by the joint probability distribution of $(\epsilon_n, \hat{\epsilon}_n)$.

Of particular interest is the quantity $\hat{\epsilon}_n - \epsilon_n$, called the *deviation*. The distribution of this random variable is called the *deviation distribution*.


bias
$P(\hat{\epsilon}_n - \epsilon_n \leq -\tau)$
variance
$\hat{\epsilon}_n - \epsilon_n$
$-\tau$
0
τ

Error Estimator Performance

Of interest in the analysis of performance of $\hat{\epsilon}_n$ are

- The *bias*,

$$\text{Bias}(\hat{\epsilon}_n) = E[\hat{\epsilon}_n] - E[\epsilon_n]$$

- The *deviation variance*,

$$\text{Var}_\text{d}(\hat{\epsilon}_n) = \text{Var}(\hat{\epsilon}_n - \epsilon_n) = \text{Var}(\hat{\epsilon}_n) + \text{Var}(\epsilon_n) - 2\text{Cov}(\hat{\epsilon}_n, \epsilon_n)$$

- The *root mean-square error*,

$$\text{RMS}(\hat{\epsilon}_n) = \sqrt{E[(\hat{\epsilon}_n - \epsilon_n)^2]} = \sqrt{\text{Var}_\text{d}(\hat{\epsilon}_n) + \text{Bias}(\hat{\epsilon}_n)^2}$$

(this combines $\text{Bias}(\hat{\epsilon}_n)$ and $\text{Var}_\text{d}(\hat{\epsilon}_n)$ in one measure)

- The tail probabilities,

$$P(\hat{\epsilon}_n - \epsilon_n \geq \tau) \text{ and } P(\hat{\epsilon}_n - \epsilon_n \leq -\tau), \quad \text{for } \tau > 0$$

Some Observations

- Note that $\text{Bias}(\hat{\epsilon}_n)$, $\text{Var}_\text{d}(\hat{\epsilon}_n)$, and $\text{RMS}(\hat{\epsilon}_n)$ are respectively the mean, the variance, and the square root of the second moment of the distribution of $\hat{\epsilon}_n - \epsilon_n$ (i.e., the deviation distribution).
- For best error estimator performance, we want the parameters $\text{Bias}(\hat{\epsilon}_n)$, $\text{Var}_\text{d}(\hat{\epsilon}_n)$, and $\text{RMS}(\hat{\epsilon}_n)$ to be as small as possible.
- If ϵ_n does not change much for different S_n (this is usually true if n is not too small), i.e. if $\epsilon_n \approx E[\epsilon_n]$, then

$$\text{Var}(\epsilon_n) = E\left[(\epsilon_n - E[\epsilon_n])^2\right] \approx 0$$

$$\text{Cov}(\hat{\epsilon}_n, \epsilon_n) = E\left[(\hat{\epsilon}_n - E[\hat{\epsilon}_n])(\epsilon_n - E[\epsilon_n])\right] \approx 0$$

so that $\text{Var}_\text{d}(\hat{\epsilon}_n) \approx \text{Var}(\hat{\epsilon}_n)$

Consistency

- Given a classification rule, an error estimator ϵ_n is said to be consistent (resp. strongly consistent) if

$$\hat{\epsilon}_n \to \epsilon_n \text{ as } n \to \infty$$

in probability (resp. with probability one).

- Clearly, consistency has to do with the tail probabilities.
    - $\hat{\epsilon}_n$ is consistent if and only if, for all $\tau > 0$

$$P(|\hat{\epsilon}_n - \epsilon_n| \geq \tau) \to 0$$

    - $\hat{\epsilon}_n$ is strongly consistent if, for all $\tau > 0$

$$P(|\hat{\epsilon}_n - \epsilon_n| \geq \tau) \to 0 \text{ and } \sum_{n=1}^{\infty} P(|\hat{\epsilon}_n - \epsilon_n| \geq \tau) < \infty$$

Estimation of Bayes Error

- If the given classification rule is consistent and the error estimator is consistent, then we have

$$\left.\begin{array}{l} \hat{\epsilon}_n \to \epsilon_n \\ \epsilon_n \to \epsilon^* \end{array}\right\} \Rightarrow \quad \hat{\epsilon}_n \to \epsilon^*$$

  so that the error estimator can be used in principle to estimate the Bayes error with a large data sample S_n.
- Convergence of $\hat{\epsilon}_n$ to ϵ_n can be shown to be fast for some classification rules and error estimators regardless of the distribution (more on this later).
- However, there will always distributions for which ϵ_n converges to ϵ^* arbitrarily slowly (DGL Thm 7.2). Therefore, one cannot guarantee that $\hat{\epsilon}_n$ is close to ϵ^* for a given n, unless one has additional information about the distribution.

The Test-Set Estimator

Here we assume that there is a set of *testing data* $S_m = \{(x_i^t, y_i^t); i = 1, \ldots, m\}$, which is *not used* in classifier design, and we define

$$\hat{\epsilon}_{n,m} = \frac{1}{m} \sum_{i=1}^{m} |y_i^t - \psi_n(x_i^t)|$$

Since S_m is random and independent from the training data, this is a randomized error estimator.

The Test-set Estimator - II

The estimator $\hat{\epsilon}_{n,m}$ has many nice properties.

- It is unbiased: $E[\hat{\epsilon}_{n,m}|S_n] = \epsilon_n \;\Rightarrow\; E[\hat{\epsilon}_{n,m}] = E[\epsilon_n]$
- Given S_n (so that ϵ_n is a fixed parameter), $m\hat{\epsilon}_{n,m}$ is binomially distributed with parameters (m, ϵ_n):

$$P(m\hat{\epsilon}_{n,m} = k|S_n) = \binom{m}{k}\epsilon_n^k(1-\epsilon_n)^{m-k}, \quad k = 0, \ldots, m$$

- From the variance of the binomial it follows that

$$V_{\text{int}} = E[(\hat{\epsilon}_{n,m} - \epsilon_n)^2|S_n] = \frac{1}{m^2}E[(m\hat{\epsilon}_{n,m} - m\epsilon_n)^2|S_n]$$
$$= \frac{1}{m^2}m\epsilon_n(1-\epsilon_n) = \frac{\epsilon_n(1-\epsilon_n)}{m}$$

The Test-set Estimator - III

- From the preceding expression we immediately get a bound on the internal variance of the holdout estimator:

$$V_{\text{int}} \leq \frac{1}{4m}$$

which tends to zero as $m \to \infty$.

- The full variance is simply

$$\text{Var}(\hat{\epsilon}_{n,m}) = E[V_{\text{int}}] + \text{Var}[\epsilon_n].$$

Thus, for large m (so V_{int} is small), $\text{Var}(\hat{\epsilon}_{n,m}) \approx \text{Var}[\epsilon_n]$ (which is usually small, particularly for large n).

Problem with the Test-Set Estimator

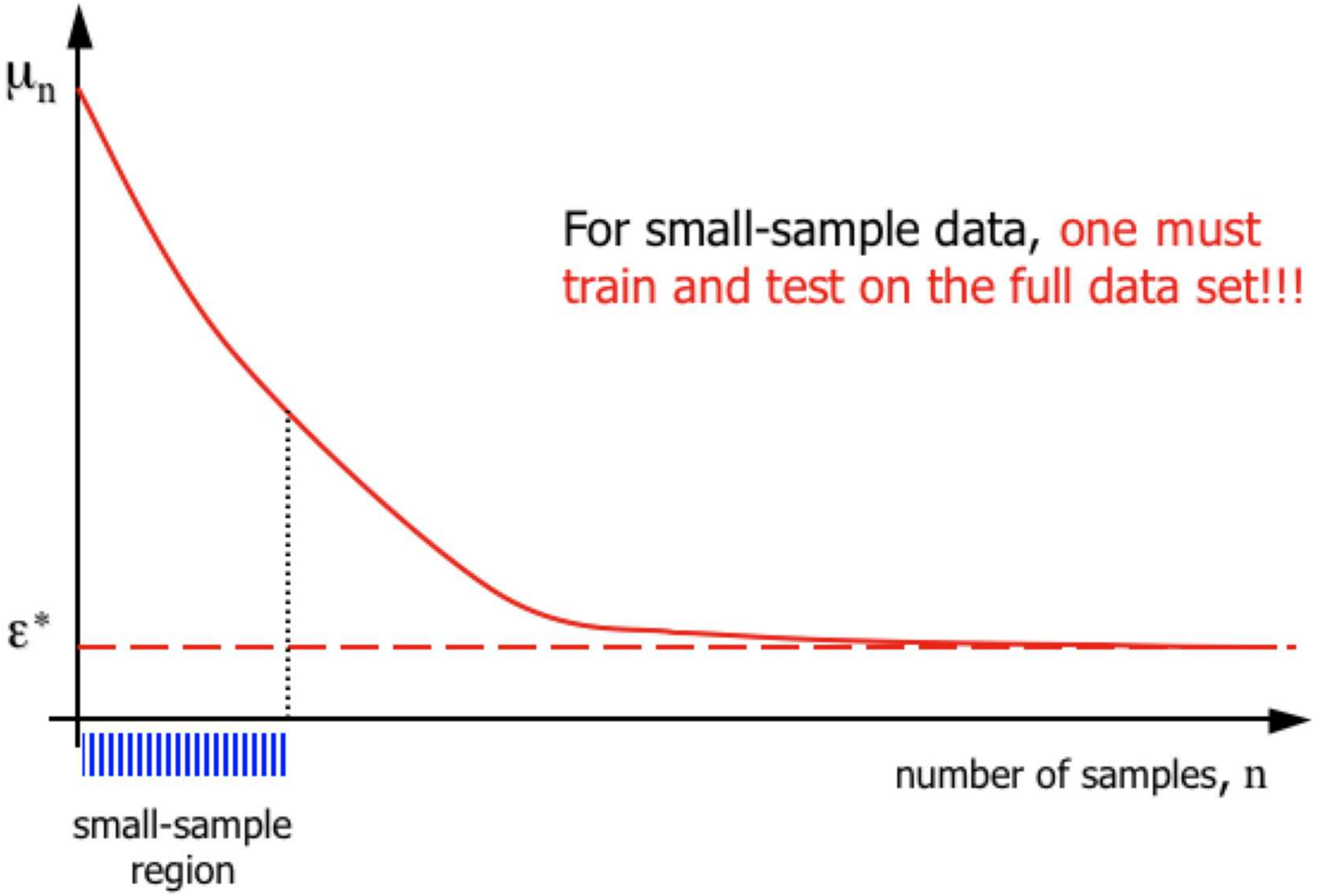
Despite its many nice properties, the test-set estimator has a serious drawback.

In practice, one has to split the available data S_n into samples for training S_{n-k} and samples for testing S_k.

Hence, one has $E[\hat{\epsilon}_{n-k,k}] = E[\epsilon_{n-k}]$, which is usually larger than $E[\epsilon_n]$, so that the estimator is pessimistically biased in this sense. If the bias is small, there is no problem. But in *small-sample* cases, the bias can be large.

Of perhaps more concern is the variance. If the number k of testing samples is small, then the variance of $\hat{\epsilon}_{n-k,k}$ is usually large (since the internal variance can be large).

Small-sample Problem



Data-Efficient Error Estimators

We will discuss the following error estimators, for which all of the training data is used to both design the classifier and estimate its future performance.

- Resubstitution (apparent error)
- Cross-Validation
- Bootstrap-based error estimators
- Bolstered error estimators

Resubstitution

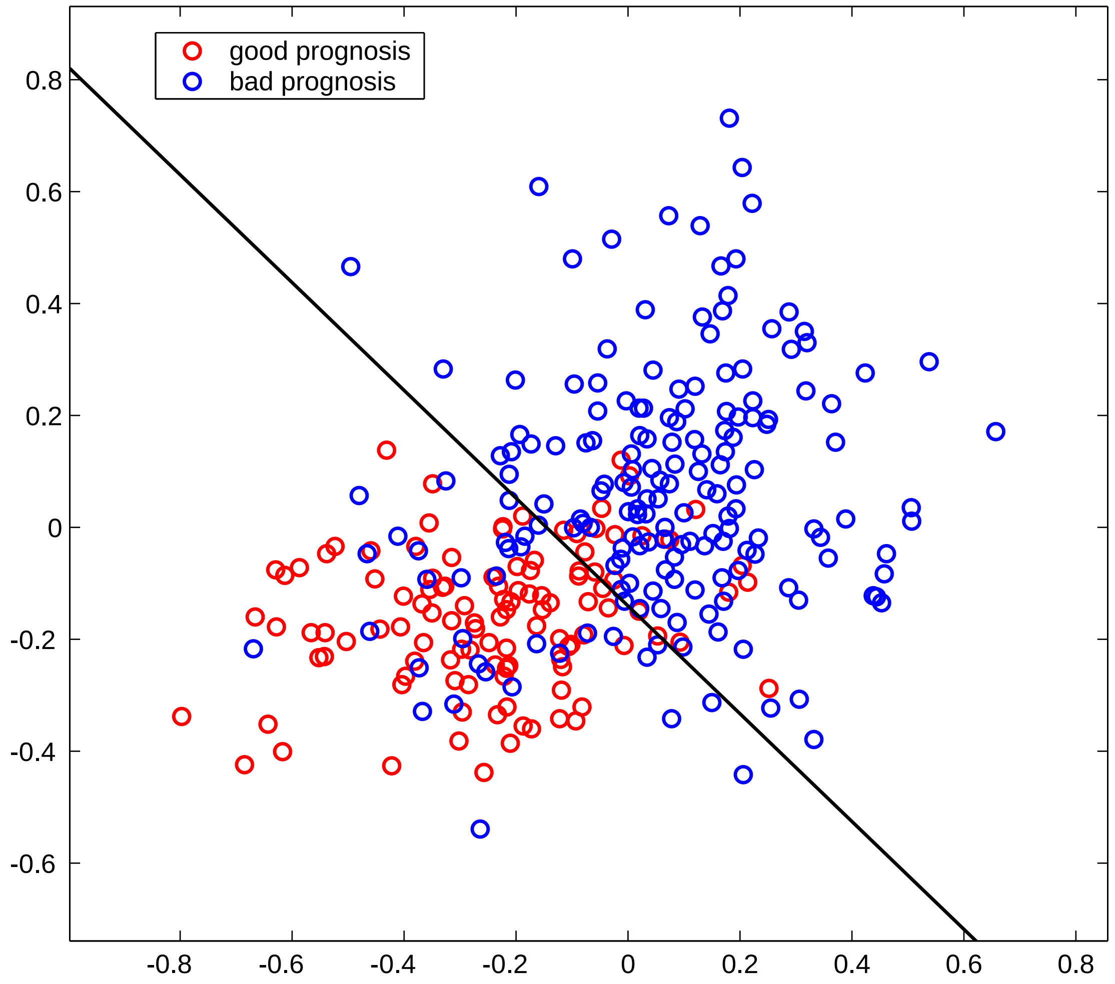
- Resubstitution is the simplest alternative. It is simply the *apparent error*, or the error on the training data:

$$\hat{\epsilon}_n^r = \frac{1}{n}\sum_{i=1}^{n} |y_i - \psi_n(x_i)|$$

- Its advantages are that it is a non-randomized estimator and the low computational complexity. It is lightning fast and so attractive in applications with large data sets.

- Its biggest drawback is that it is *usually* optimistically biased, $E[\hat{\epsilon}_n^r] < E[\epsilon_n]$. The bias tends to be larger for complex classification rules (due to overfitting). As an extreme example, we have the 1-NN rule, for which $\hat{\epsilon}_n^r \equiv 0$, regardless of the data.

Example



$$\hat{\epsilon}_n^r = \frac{\text{errors committed}}{\text{number of points}}$$

$$= \frac{52}{295} = 17.6\%$$

Histogram Classification

(Thm 23.3 DGL – Distribution-free bounds for resubstitution for histogram classifiers)

For a histogram classifier, we have that

$$\text{Var}(\hat{\epsilon}_n^r) \leq \frac{1}{n}$$

Resubstitution is optimistically-biased,

$$\text{Bias}(\hat{\epsilon}_n^r) \leq 0 \Rightarrow E[\hat{\epsilon}_n^r] \leq E[\epsilon_n]$$

In fact, resubstitution is *really* biased: $E[\hat{\epsilon}_n^r] \leq \epsilon^* \leq E[\epsilon_n]$.

For finite number of bins b (e.g., discrete histogram rule),

$$\text{RMS}(\hat{\epsilon}_n^r) \leq \sqrt{\frac{6b}{n}}$$

General Linear Discriminants

(Thm 23.1 DGL – Distribution-free bound for tail probabilities of resubstitution for general linear discriminants)

For a classification rule such that

$$\psi_n(x) = \begin{cases} 1, & a_{0,n} + \sum_{i=1}^{K} a_{i,n}\varphi_i(x) \geq 0 \\ 0, & \text{otw} \end{cases}$$

For all n and $\tau > 0$, we have that

$$P(|\hat{\epsilon}_n^r - \epsilon_n| \geq \tau) \leq 8n^K e^{-n\tau^2/32}$$

In particular, $\hat{\epsilon}_n^r$ is strongly consistent, and convergence of $\hat{\epsilon}_n^r$ to ϵ_n is fast (exponential) for any distribution.

Cross-Validation

- Cross-validation removes the optimism from resubstitution by employing test points not used in classifier design.
- In k-fold cross-validation, S_n is partitioned into k *folds* $S_{(i)}$, for $i = 1, \ldots, k$ (assume that k divides n). Each fold contains n/k samples that are left out of training and used as a test set. The process is repeated k times (once for each fold) and the estimate is the overall proportion of errors committed on all folds:

$$\hat{\epsilon}_n^{cv(k)} = \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{n/k} |y_j^{(i)} - \Psi_{n-n/k}(x_j^{(i)}; S_n \backslash S_{(i)})|,$$

where $(x_j^{(i)}, y_j^{(i)})$ is a sample in the i-th fold.

Cross-Validation - II

- The process can be repeated by using different partitions and averaging the results (the averaging helps to reduce the internal variance).
- This is a randomized error estimator (why?) and can be a slow estimator for large n and k.
- Cross-validation is advertized as unbiased. But it is unbiased as an estimator of $\epsilon_{n-n/k}$ (provided the folds are picked randomly):

  $$E[\hat{\epsilon}_n^{cv(k)}] = E[\epsilon_{n-n/k}]$$

  Usually, this means that it is *pessimistically* biased as an estimator of ϵ_n.
- The most important drawback of cross-validation however is its large variability on small sample sets.

Leave-one-out Error Estimator

- The *leave-one-out* error estimator corresponds to n-fold cross-validation, whereby a single observation is left out each time:

$$\hat{\epsilon}_n^{\,l} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \Psi_{n-1}(x_i; S_{n-1}^{i})|,$$

  where S_{n-1}^{i} is the data set resulting from deleting data point i from the original data set S_n.

- It is unbiased as an estimator of ϵ_{n-1}: $E[\hat{\epsilon}_n^{\,l}] = E[\epsilon_{n-1}]$
- This is a non-randomized estimator!

Surrogate Classifiers

- Note the following curious fact: the designed classifier ψ_n is not used to compute $\hat{\epsilon}_n^{cv(k)}$, but rather "surrogate" classifiers

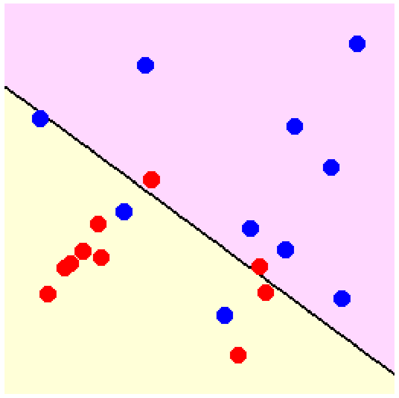
$$\psi^i_{n-n/k} = \Psi_{n-n/k}(\cdot\,; S_n \backslash S_{(i)}) \quad i = 1\ldots,k$$

- This adds variance for unstable (complex) rules due to overfitting.

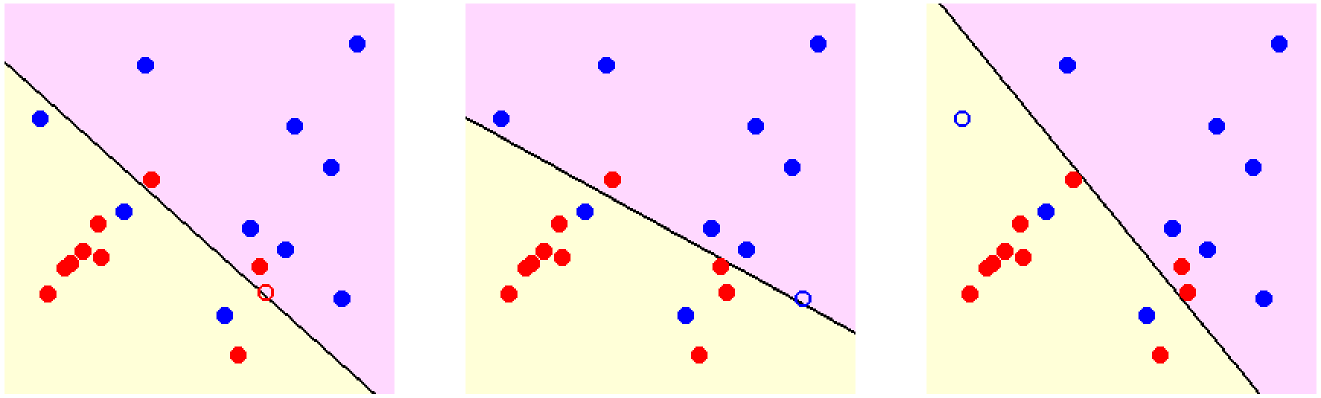
- It also makes $\hat{\epsilon}_n^{cv(k)}$ an approximation of the *expected* classification error $E[\epsilon_{n-n/k}]$ rather than ϵ_n or $\epsilon_{n-n/k}$.

Surrogate Classifiers - LDA Example

Original LDA classifier

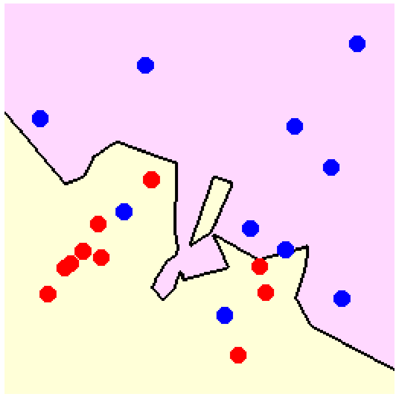


A few surrogate LDA classifiers

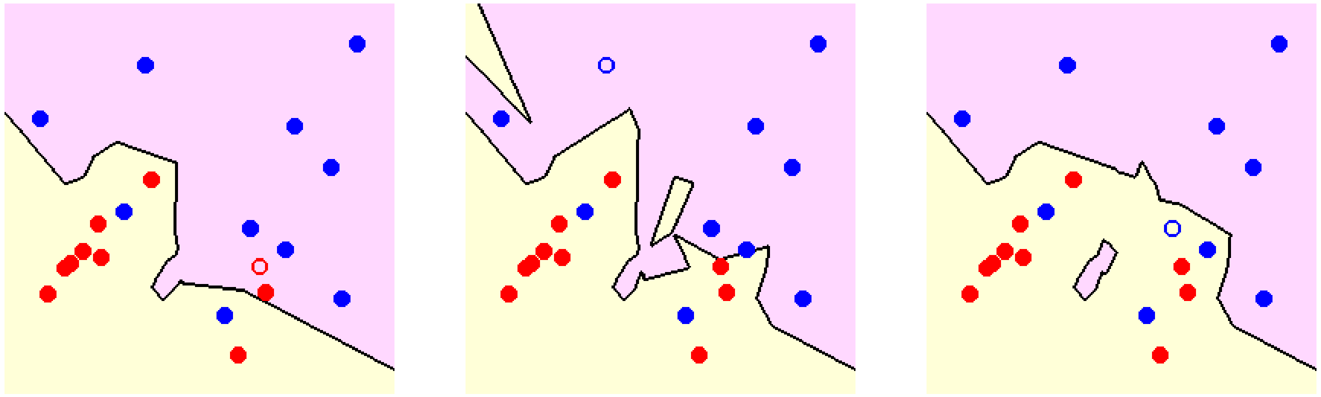


Surrogate Classifiers - 3NN Example

Original 3NN classifier

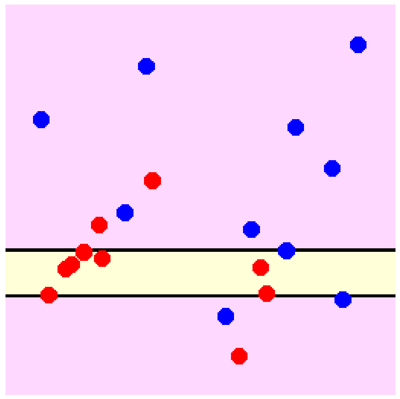


A few surrogate 3NN classifiers

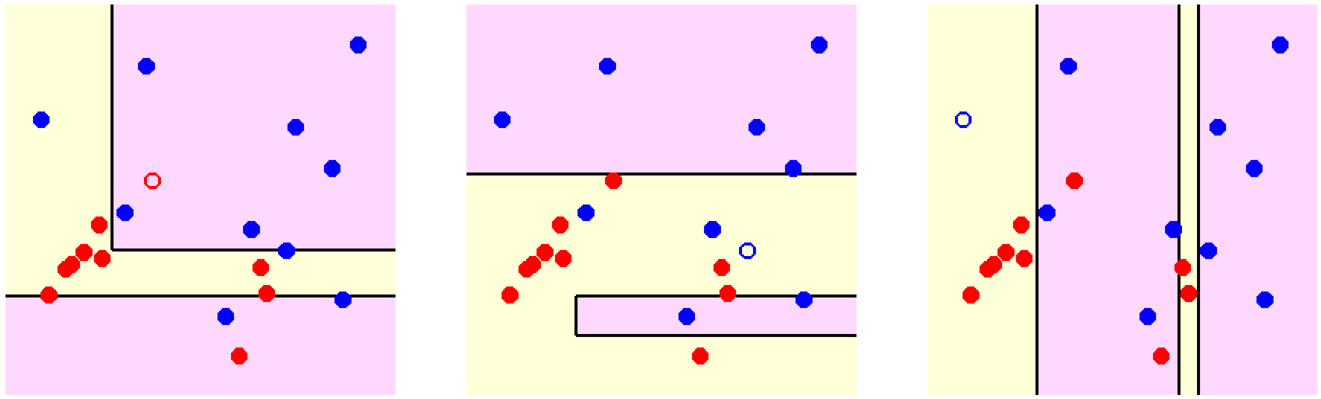


Surrogate Classifiers - CART Example

Original CART classifier



A few surrogate CART classifiers



Histogram Classification Rule

(Thm 24.7 DGL – Distribution-free bounds for leave-one-out for histogram classifiers)

For a histogram classifier, we have that

$$\text{RMS}(\hat{\epsilon}_n^l) \leq \sqrt{\frac{1+6/e}{n} + \frac{6}{\sqrt{\pi(n-1)}}}$$

Note that this bound, which can be shown to be tight for some distributions, decreases much more slowly with increasing n than the one for resubstitution with a finite number of bins.

General Upper Bound for LOO

(Thm 24.2 DGL – Distribution-free bound for RMS of leave-one-out for symmetric classification rules)

For a symmetric classification rule, i.e., a rule for which the order of the training points does not matter,

$$\text{RMS}(\hat{\epsilon}_n^l) \;\leq\; \sqrt{\frac{1}{n} + 6P\left[\Psi_n(X;S_n) \neq \Psi_{n-1}(X;S_{n-1})\right]}$$

Note that the probability of the surrogate classifiers differing from the original classifier directly influences the bound.

Bootstrap

- Define the *empirical distribution* of the data as the discrete probability mass function P_n on $R^d \times \{0,1\}$:

$$P_n(x,y) = \begin{cases} \frac{1}{n}, & x = x_i,\ y = y_i \\ 0, & \text{otw} \end{cases}$$

  This puts equal mass $\frac{1}{n}$ at the observed data points.

- What happens when we sample from *this* distribution rather than the original true distribution of the data? This is the idea behind the bootstrap method.

- A *bootstrap sample* S_n^b is a random sample of size n from P_n; it consists of n equally-likely draws with replacement from the original data S_n. Some of the original training samples may appear multiple times in S_n^b, whereas others may not appear at all.

Bootstrap - II

- The basic bootstrap *zero* error estimator consists of testing on the samples left out of the bootstrap sample, and averaging over several bootstrap samples.

- This is similar to cross-validation, but here the classification rule operates on n sample points.

- Ideally, the estimator can be written as the expected value of this procedure (i.e., an average over of an infinite number of bootstrap samples):

$$\hat{\epsilon}_0 = E\left[|Y - \Psi_n(X; S_n^b)| \,\middle|\, S_n, (X,Y) \in S_n \setminus S_n^b\right]$$

Bootstrap - III

- In practice, this is approximated by a a Monte-Carlo estimate based on a number of bootstrap samples $S_n^{b(i)}$, for $i = 1, \ldots, B$ (B between 25 and 200 being typical):

$$\hat{\epsilon}_0 \approx \frac{1}{K} \sum_{i=1}^{B} \sum_{j=1}^{n} |y_j - \Psi_n(x_j; S_n^{b(i)})| \, I_{(x_j,y_j) \in S_n \setminus S_n^{b(i)}}$$

where

$$K = \sum_{i=1}^{B} \sum_{j=1}^{n} I_{(x_j,y_j) \in S_n \setminus S_n^{b(i)}}$$

is the total number of bootstrap test samples.

- This MC estimate yields a randomized error estimator. Its internal variance is the variance of the sample mean, which has to be made small by using large B.

The 0.632 Bootstrap

- Like cross-validation, the bootstrap estimator $\hat{\epsilon}_0$ will be in general pessimistically biased as an estimator of ϵ_n, since the amount of distinct samples available for designing the classifier is on average only

$$(1 - e^{-1})n \approx 0.632n < n$$

- The 0.632 bootstrap error estimator

$$\hat{\epsilon}_{\mathrm{b632}} = (1 - 0.632)\, \hat{\epsilon}_n^{\,r} + 0.632\, \hat{\epsilon}_0 .$$

tries to correct this bias by averaging with the (usually) optimistically-biased resubstitution.

The 0.632 Bootstrap - II

- The 0.632 bootstrap has been widely considered the best-performing error estimator in machine learning and data mining.
- It has small variance and bias, but can be extremely slow to compute.
- It can fail when resubstitution is too optimistically-biased (e.g., the 1-NN rule). A variant called 0.632+ bootstrap has been proposed to address this problem.
- Other bootstrap error estimator variants include the bias-corrected bootstrap, double bootstrap, etc.

Bolstered Error Estimation

- This is an approach that achieves a reasonable compromise to the bias/variance/complexity trillema.
- It is based on the idea of modifying ("bolstering") the empirical distribution of the data.
- We will focus on bolstered resubstitution. This is generally a very fast error estimator, since, like resubstitution, it does not rely on resampling the data and designing surrogate classifiers.
- Bolstered resubstitution for linear classifiers (LDA, perceptron, linear SVM, etc.) is particularly nice, since the necessary integrations can be computed exactly, yielding a non-randomized estimator that is almost as fast as resubstitution.

Another Look at Resubstitution

Resubstitution can be written as:

$$\hat{\epsilon}_n^r = E_{P_n}\left[|Y - \psi_n(X)|\right]$$

that is, it is the classification error of ψ_n if (X,Y) were distributed according to the empirical distribution P_n.

Note that this makes no distinction between points far and near the decision boundary. Regardless, each point will contribute:

- zero if it is correctly classified
- $1/n$ if it is misclassified

This situation changes if we suitably modify the empirical distribution.

Bolstered Empirical Distribution

- Main idea: spread out the probability mass put on each point by the empirical distribution.
- Define the *bolstered empirical distribution* $F^\diamond$, with probability density function $f^\diamond$ given by:

$$f^\diamond(x,y) = \frac{1}{n}\sum_{i=1}^{n} f_i^\diamond(x - x_i) I_{y=y_i}$$

where the *bolstering kernels* $f_i^\diamond$ are multivariate density functions over R^d, for $i = 1, \ldots, n$.

Bolstered Resubstitution

- Substituting the bolstered empirical distribution in the previous expression for resubstitution yields the bolstered resubstitution error estimator

$$\hat{\epsilon}_n^{\diamond} = E_{F^\diamond}\left[|Y - \psi_n(X)|\right]$$

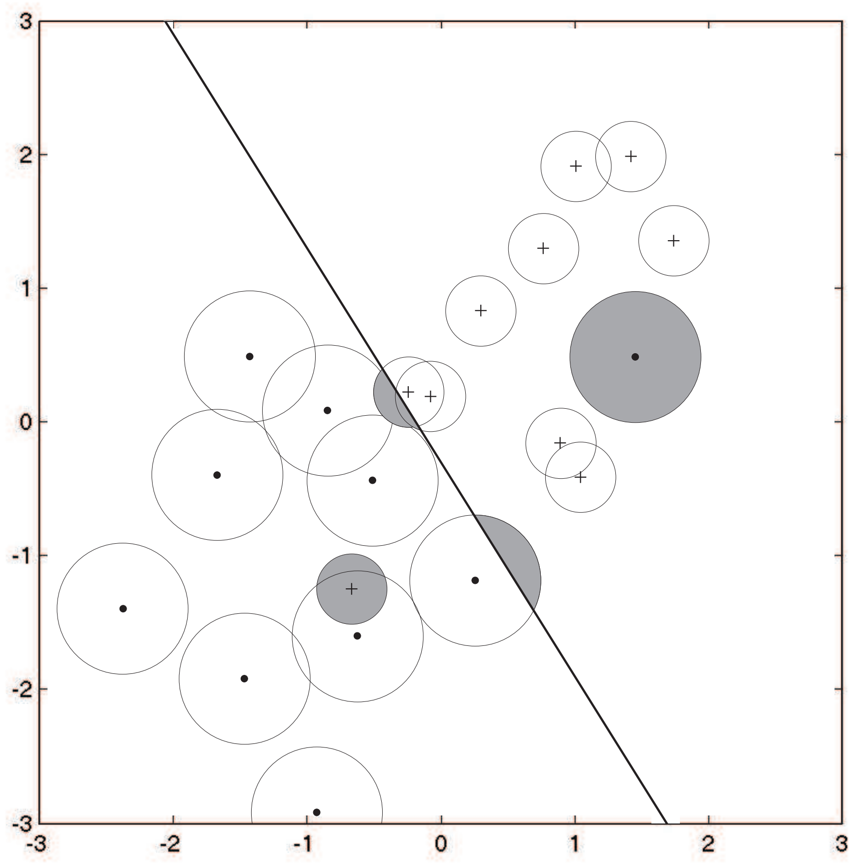
- The expectation can be written out as follows

$$\hat{\epsilon}_n^{\diamond} = \frac{1}{n}\sum_{i=1}^{n}\left(\int_{A_1} f_i^{\diamond}(x - x_i)\,dx\, I_{y_i=0} + \int_{A_0} f_i^{\diamond}(x - x_i)\,dx\, I_{y_i=1}\right)$$

where $A_j = \{x \in R^d \mid \psi_n(x) = j\}$, for $j = 0, 1$.

Example

This example illustrates the case of a linear classifier and uniform circular bolstering kernels.



Some Observations

- Points contribute to the bolstered resubstitution error according to their distance from the decision boundary, which can reduce bias and variance symultaneously.
- For linear classifiers, the integrals can usually be computed exactly. for general decision boundaries, one needs to use Monte-Carlo integration:

$$\hat{\epsilon}_n^{\diamond} \approx \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{M} I_{x_{ij} \in A_1} I_{y_i=0} + \sum_{j=1}^{M} I_{x_{ij} \in A_0} I_{y_i=1} \right)$$

  where $\{x_{ij}\}$, $j = 1, \ldots, M$, are samples drawn from the distribution $f_i^{\diamond}$.
- This yields a randomized estimator. But experiments have shown that the internal variance can be made small with as few as 10 MC samples per training point.

Error Estimator Performance

- Error estimation performance is a function of
  - classification rule
  - sample size
  - dimensionality (complexity)
  - feature-label distribution
- Given the factors above, one can compare error estimators by obtaining their bias, variance, RMS and tail probabilities.
- For some classification rules and simple error estimators (e.g., resub and loo), there exist nice analytical results, such as exact formulas or universal (distribution-free) bounds, as we have seen.
- In the general case, research has relied on simulation.

Bias/Variance/Complexity Trilemma

- A good error estimator ideally will have
  - small bias (or be unbiased)
  - small variance
  - low complexity (so it will be fast to compute).
- This is a difficult trade-off. For example:
  - Resubstitution: very fast, small variance, tends to be quite (optimistically) biased
  - Cross-validation: average speed, small bias, tends to be quite variable
  - Bootstrap: small bias, small variance, very slow
  - Bolstering: offers a compromise, small bias, variance and computational complexity

Error-Counting Estimators

- Resubstitution and cross-validation estimators change by jumps of $1/n$, which introduces variability for small n.
- Repeated cross-validation and bootstrap estimators alleviates this problem, since the jump becomes
  - $(1/n)/N$ for CV with N repetitions
  - $1/M$ for bootstrap zero estimator with M bootstrap test samples ($M >> n$)
- Bolstering avoids this problem completely (when Monte-Carlo computation is not needed), by not using error-counting at all.

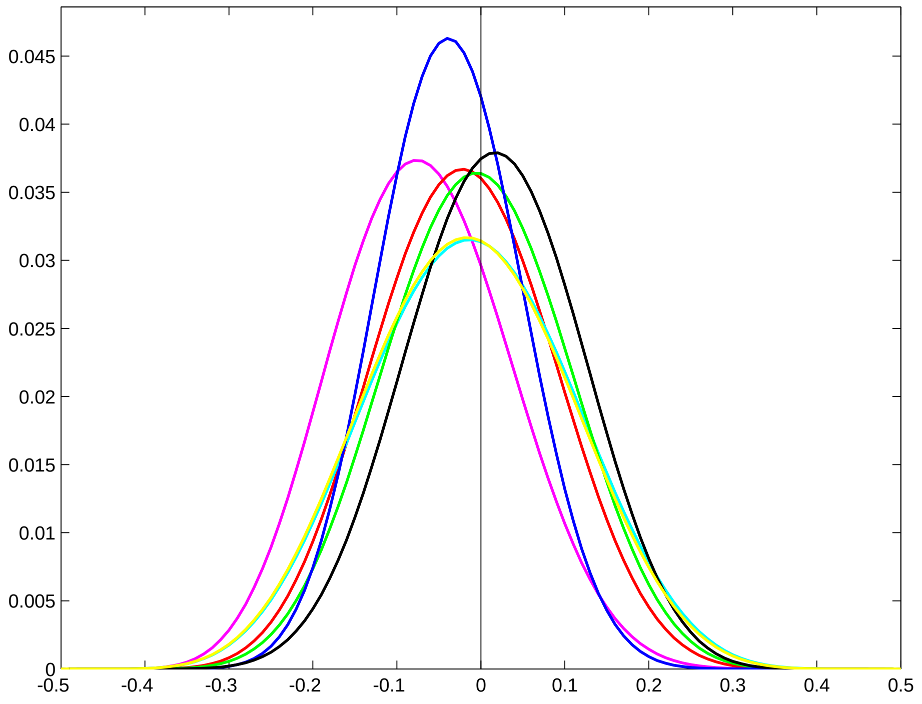
Simulation - LDA/3NN/CART

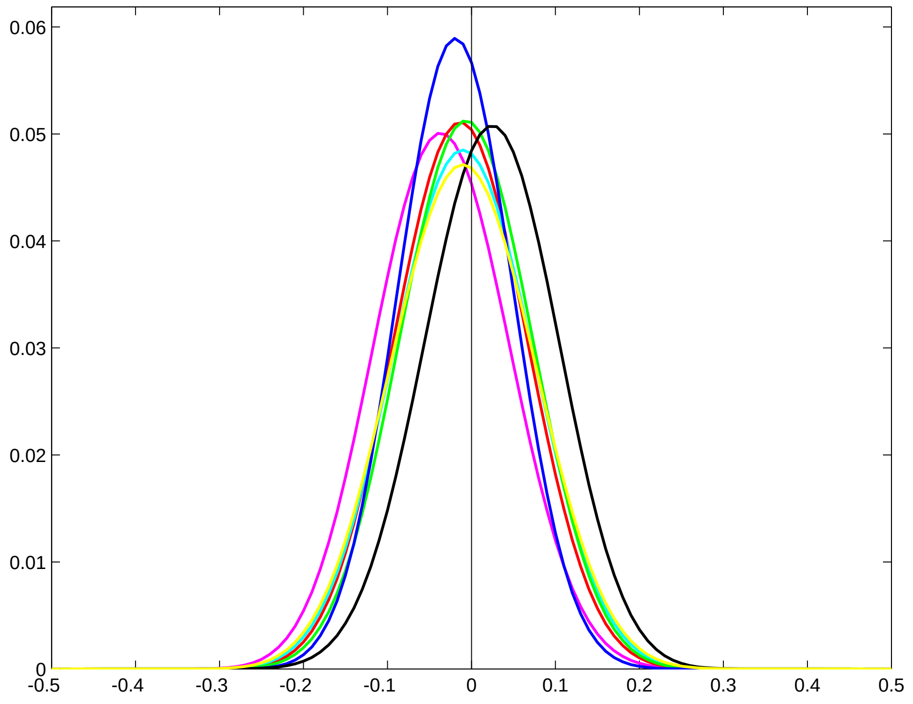
- Compute deviation distributions, using cancer data (van der Vijvner et al., NEJM, 2002).
- Draw 1000 random subsets of size n from the original 295 samples. True error was estimated in each case by hold-out using remaining $295 - n$ samples.
- Classification rules: LDA, 3NN, CART.
- Error estimators: resubstitution (resub); leave-one-out (loo); 10-fold cv with 10 repetitions (cv10r); .632 bootstrap (b632); bolstered resubstitution (bresub); semi-bolstered resubstitution (sresub); bolstered leave-one-out (bloo)
- Deviation distribution is represented by fitting a beta density to the 1000 computed values for $\hat{\epsilon}_n - \epsilon_n$.

Deviation Distributions - LDA

resub 🟪 loo 🟨 cv10r 🟦 b632 🟥 bresub 🟦 sresub ⬛ bloo 🟩

$n = 20$ $n = 40$

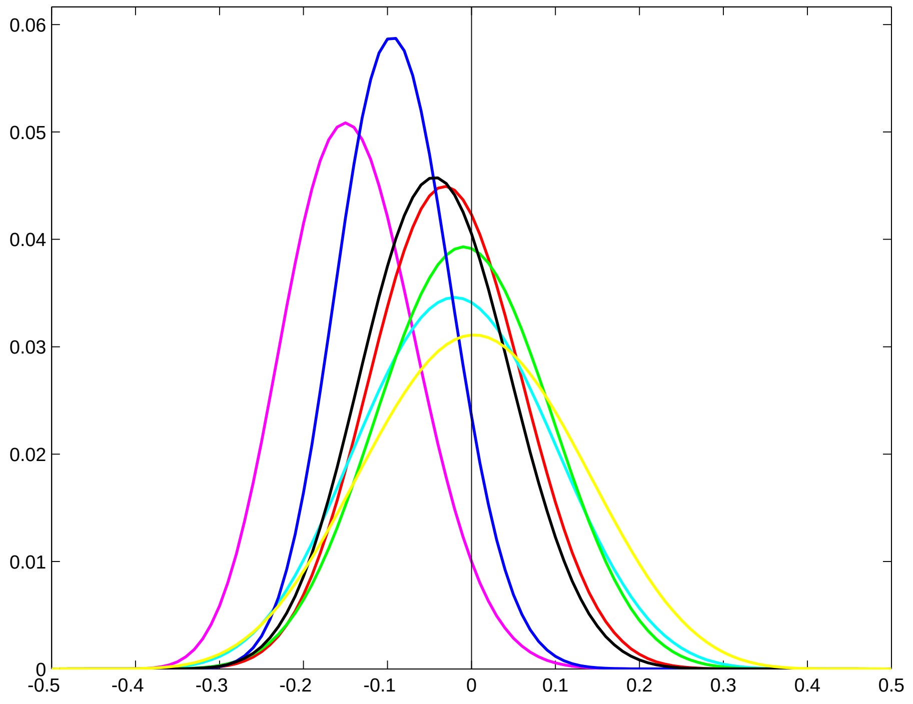




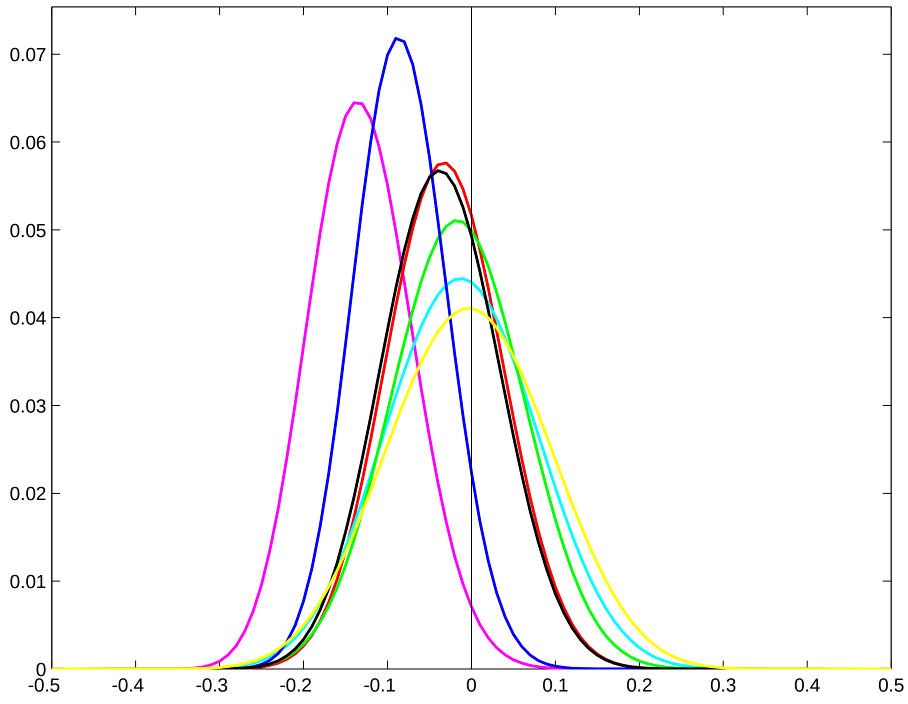
Deviation Distributions - 3NN

resub 🟪 loo 🟨 cv10r 🟦 b632 🟥 bresub 🟦 sresub ⬛ bloo 🟩

$n = 20$



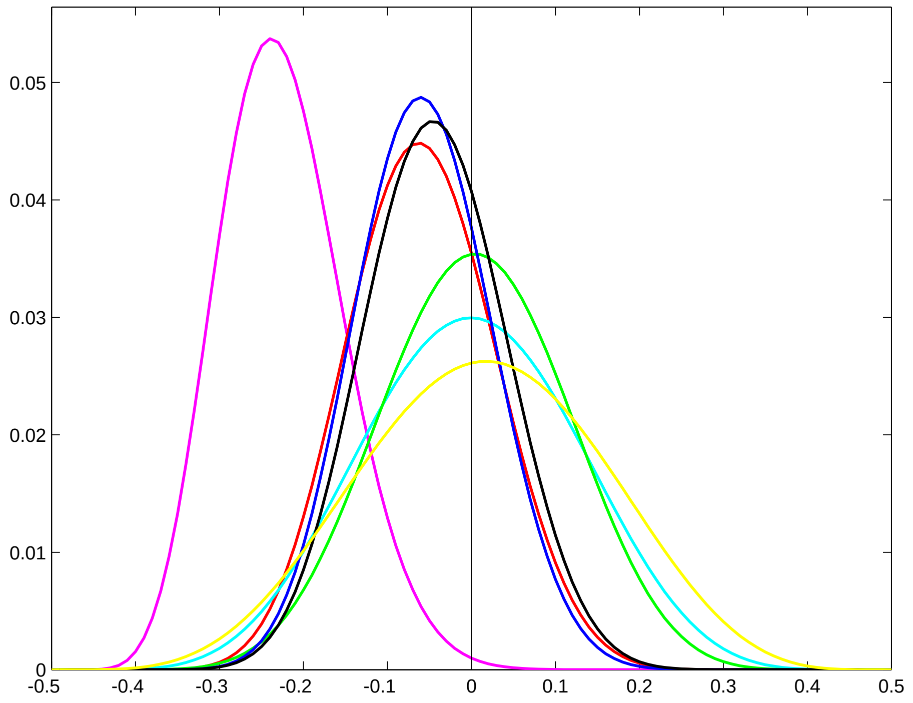
$n = 40$

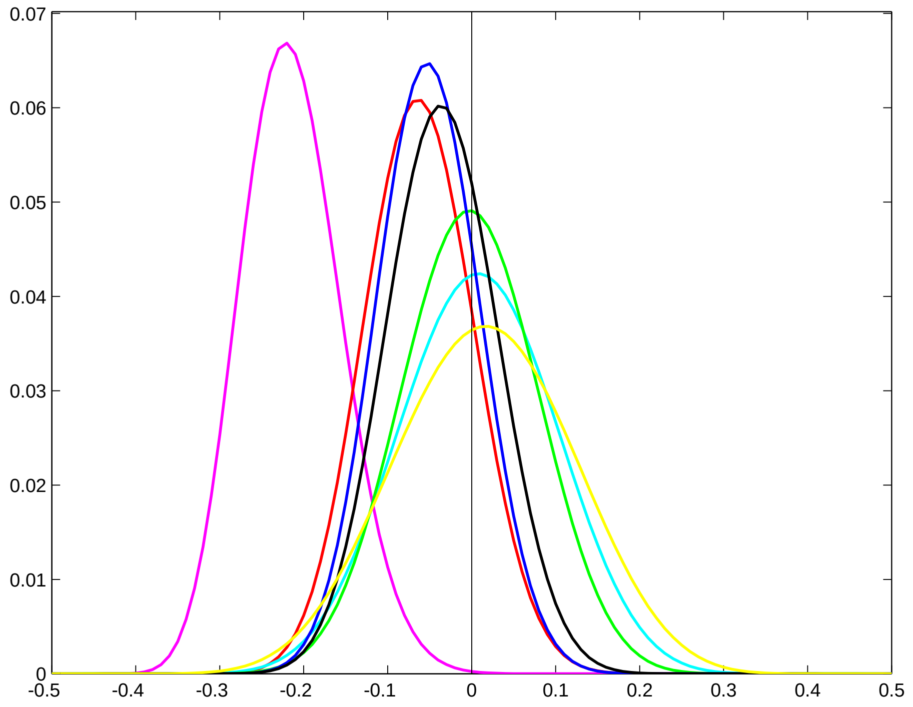


Deviation Distributions - CART

resub 🟪 loo 🟨 cv10r 🟦 b632 🟥 bresub 🟦 sresub ⬛ bloo 🟩

$n = 20$ $n = 40$





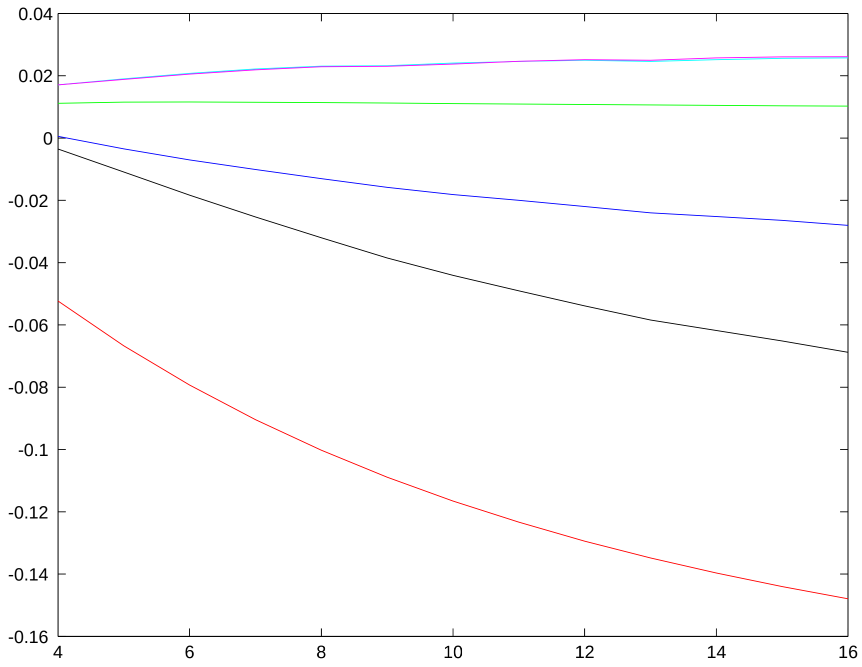
Simulation - Discrete Histogram

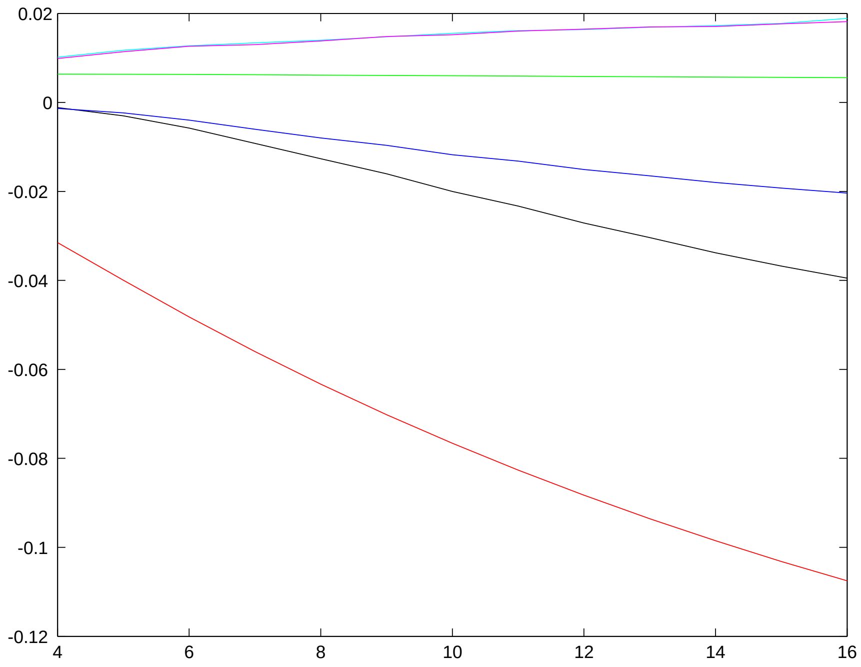
- We assume a Zipf model (same as in the discrete classification lecture)
- Error estimators: resubstitution (resub); leave-one-out (loo); 4-fold cv (cv4); 4-fold cv with 10 repetitions (cv4r); bias-corrected bootstrap (bbc); .632 bootstrap (b632);
- We computed exact results for resubstitution and leave-one-out, and Monte-Carlo approximate results for cross-validation and bootstrap.
- We display bias, variance, and RMS as a function of the number of bins (i.e., the complexity of the classifier), for expected error 20% (moderate prediction difficulty) and two sample sizes.

Bias

resub 🟥 loo 🟩 cv4 🟦 cv4r 🟪 bbc ⬛ b632 🟦

$n = 20$ $n = 40$

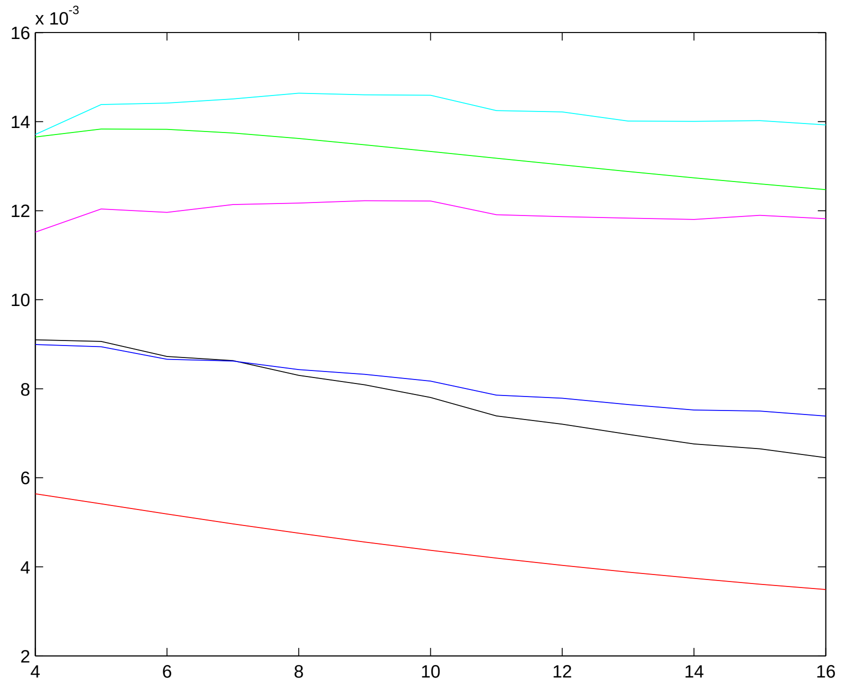


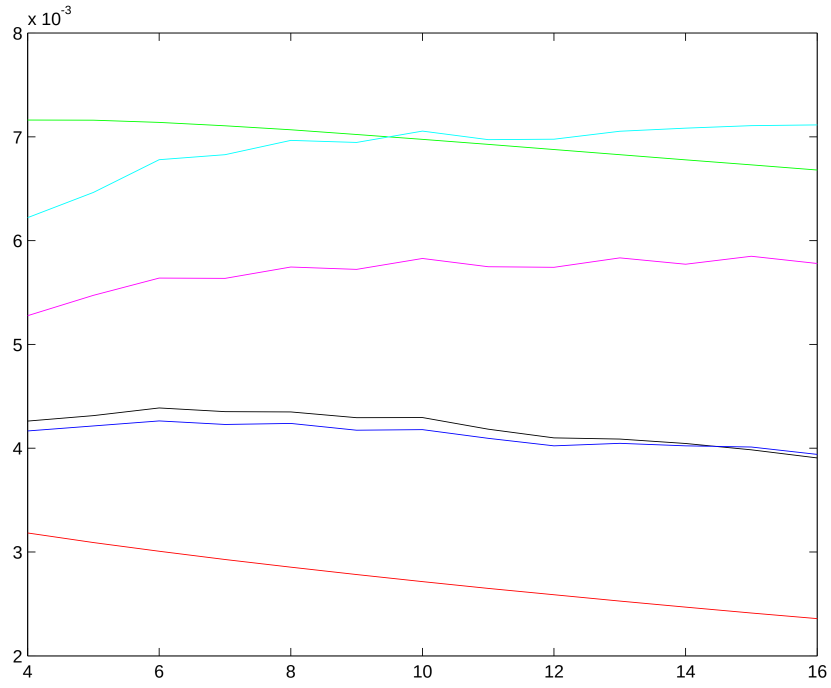


Variance

resub 🟥 loo 🟩 cv4 🟦 cv4r 🟪 bbc ⬛ b632 🟦

$n = 20$ $n = 40$

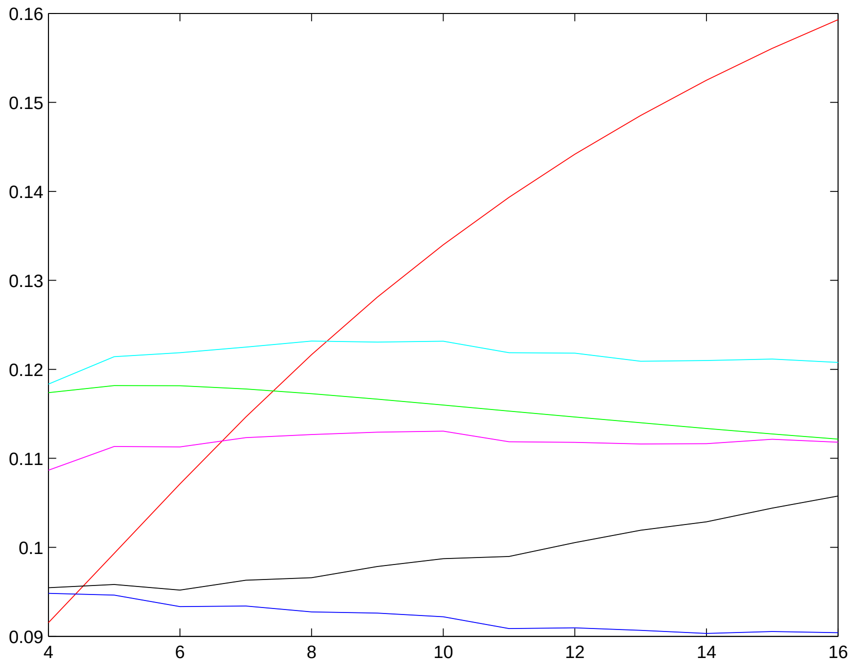


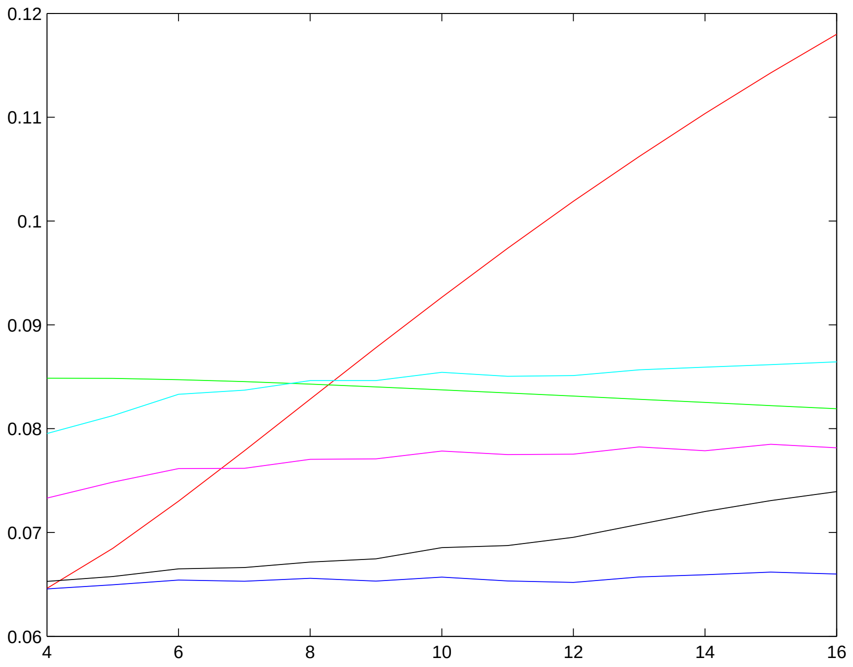


RMS

resub 🟥 loo 🟩 cv4 🟦 cv4r 🟪 bbc ⬛ b632 🟦

$n = 20$ $n = 40$





The RMS of resub is smaller than the RMS of b632 for $b = 4$!