Error Estimation for Pattern Recognition

Ulisses de Mendonça Braga Neto, Ph.D.

Texas A&M University

1. Introduction

Empirical and Mechanistic Models

- **Empirical models** are derived by observation of phenomena and model fitting.
- **Mechanistic models** are founded on basic physical principles instead.
- A good example is provided by the two models for planetary motion (both are deterministic): Kepler's empirical model vs. Newton's mechanistic model .
- However, deterministic and mechanistic modeling are largely ineffective in complex domains due to the presence of significant unexplained variability.

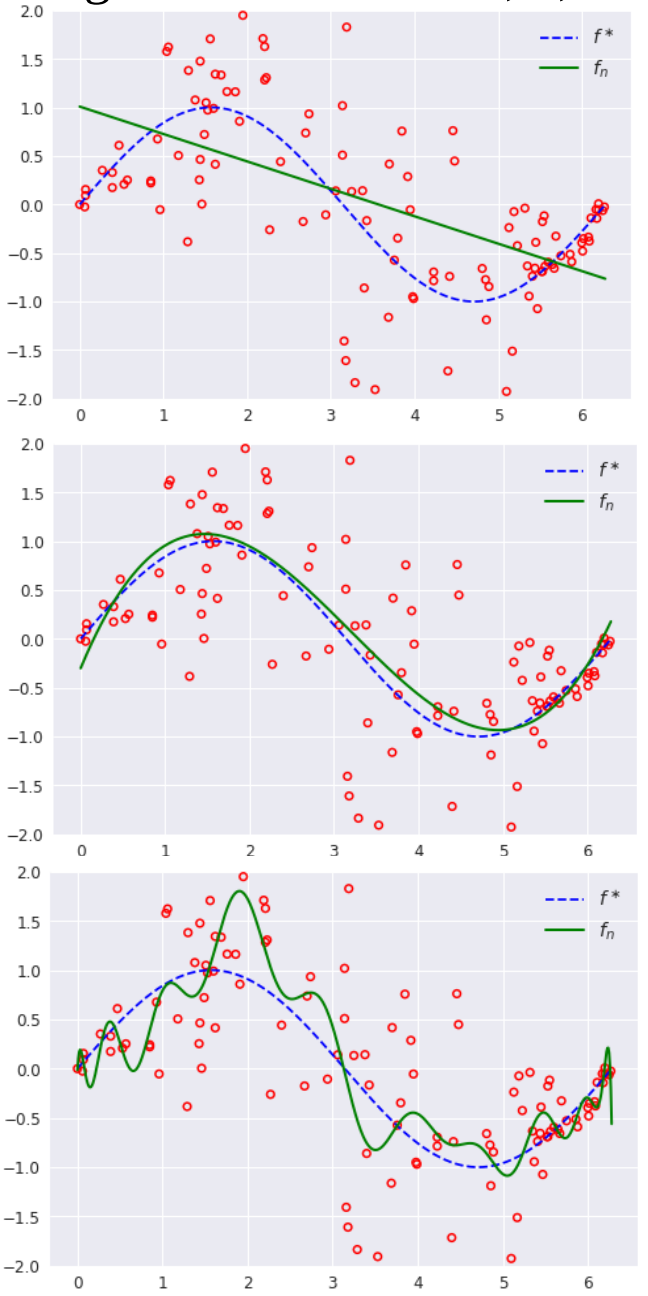
The Least-Squares Method

"The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum."

– Gauss.

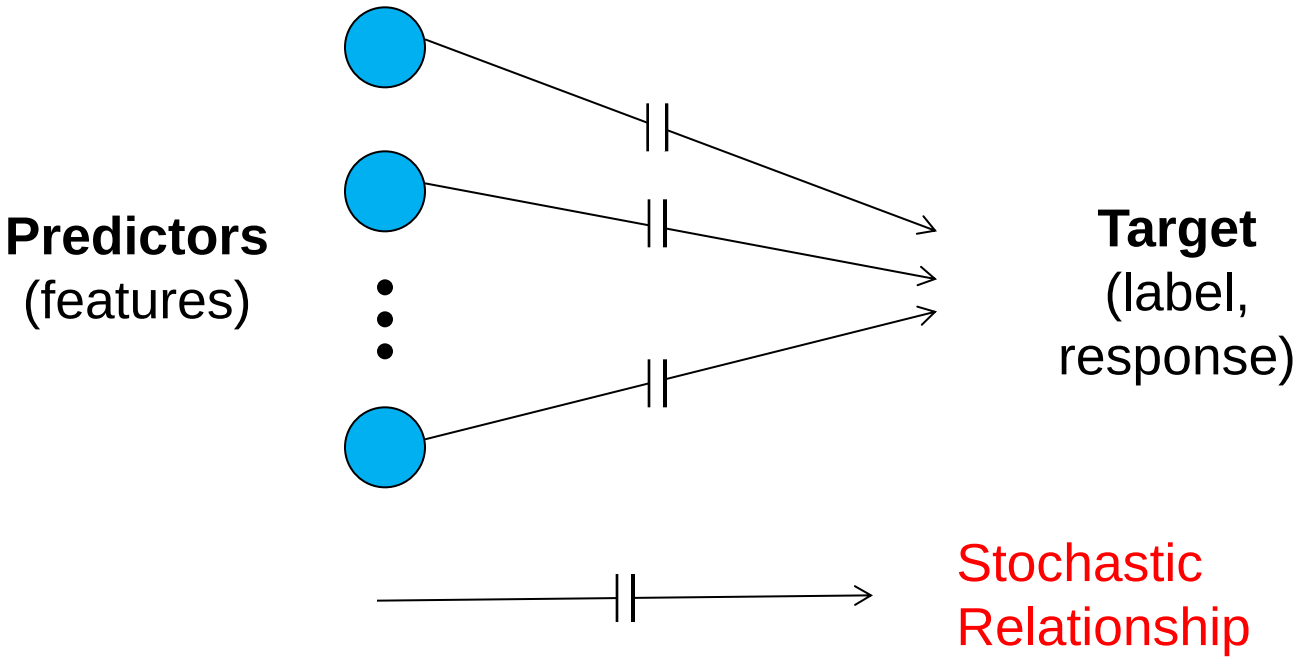
Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium, 1809

Least-squares polynomial regression of orders 1, 3, 24



Stochastic Prediction

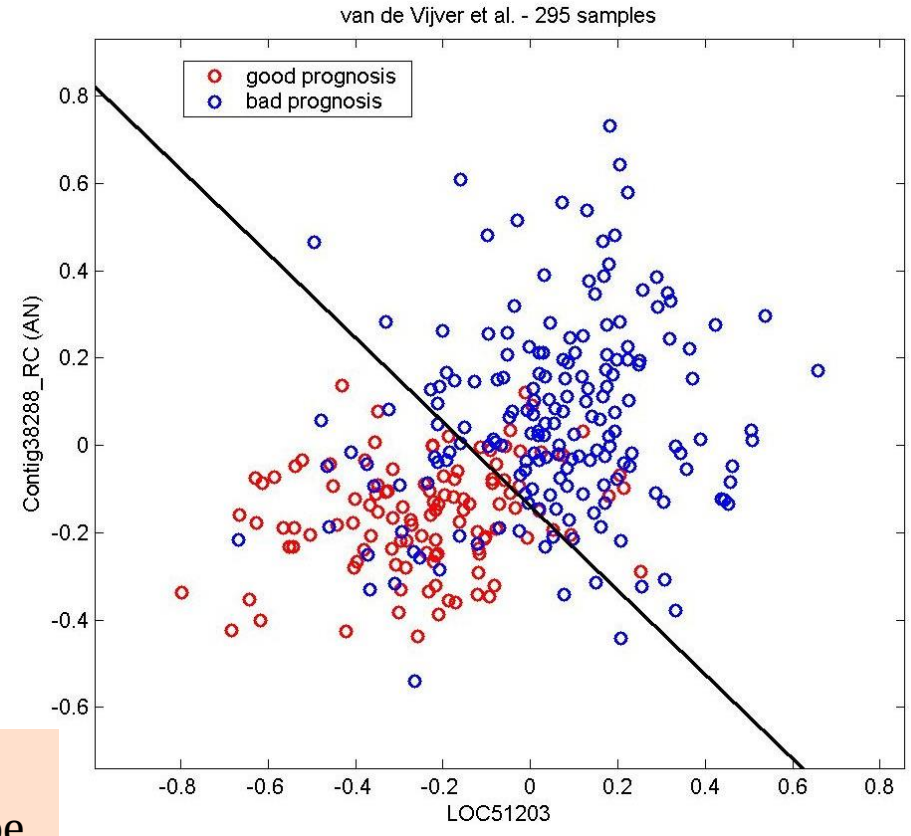
- Goal: **stochastic empirical models.**



Classifier Error Estimation

How to estimate the future performance (error rate) of this classifier?

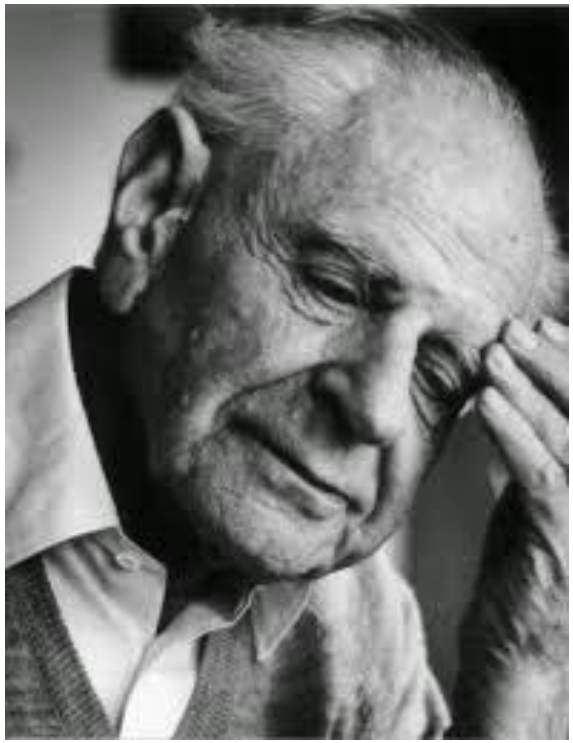
Breast Cancer Microarray Data



One of our goals will be to show under what conditions the apparent error can be an excellent error estimator.

Apparent Error = 52/295 = 17.6%

Science is Based on Testing Predictions

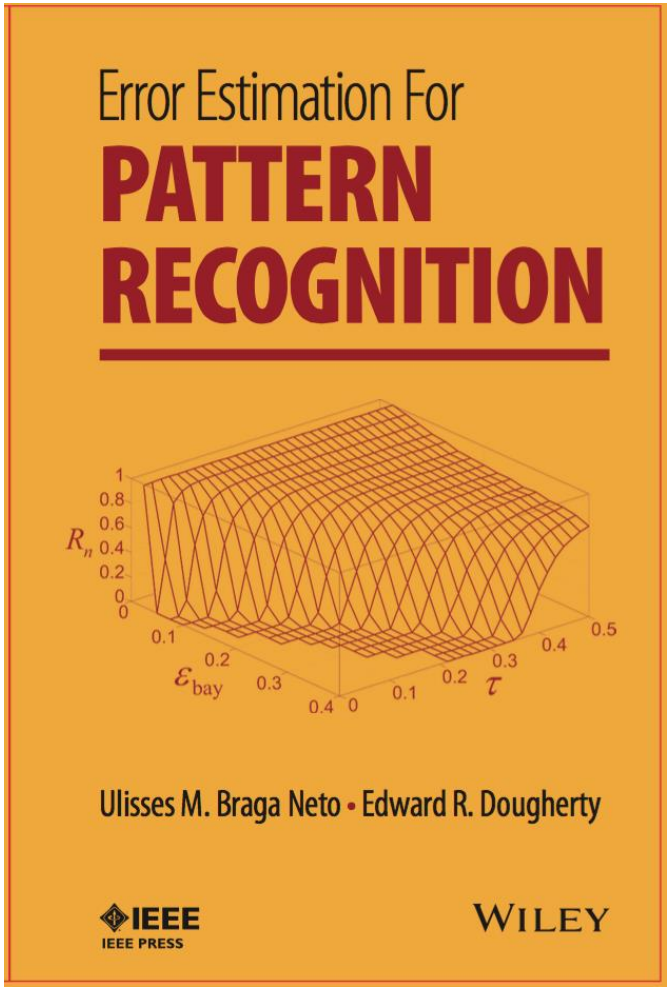


"The game of science is, in principle, without end. He who decides one day that scientific statements do not call for any further test, and that they can be regarded as finally verified, retires from the game."

– Karl Popper.

The Logic of Scientific Discovery, 1935

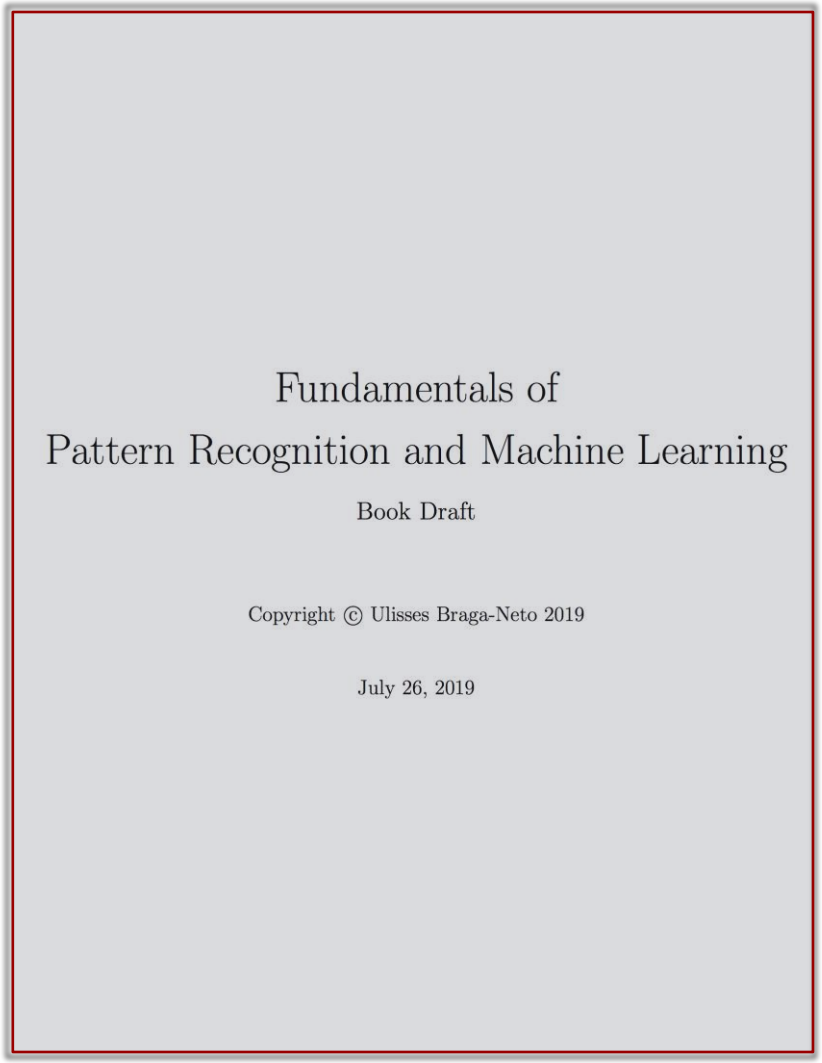
Braga-Neto and Dougherty, *Error Estimation for Pattern Recognition*, Wiley-IEEE Press, 2015.



This book is the first one dedicated exclusively to the topic of error estimation for pattern recognition. It covers both classical and recent results on the performance of error estimators for nonparametric and parametric classifiers.

Many of the issues related to Big Data discussed in this talk are covered in detail in the book.

Braga-Neto, *Fundamentals of Pattern Recognition and Machine Learning*, Springer, 2019. (In press.)

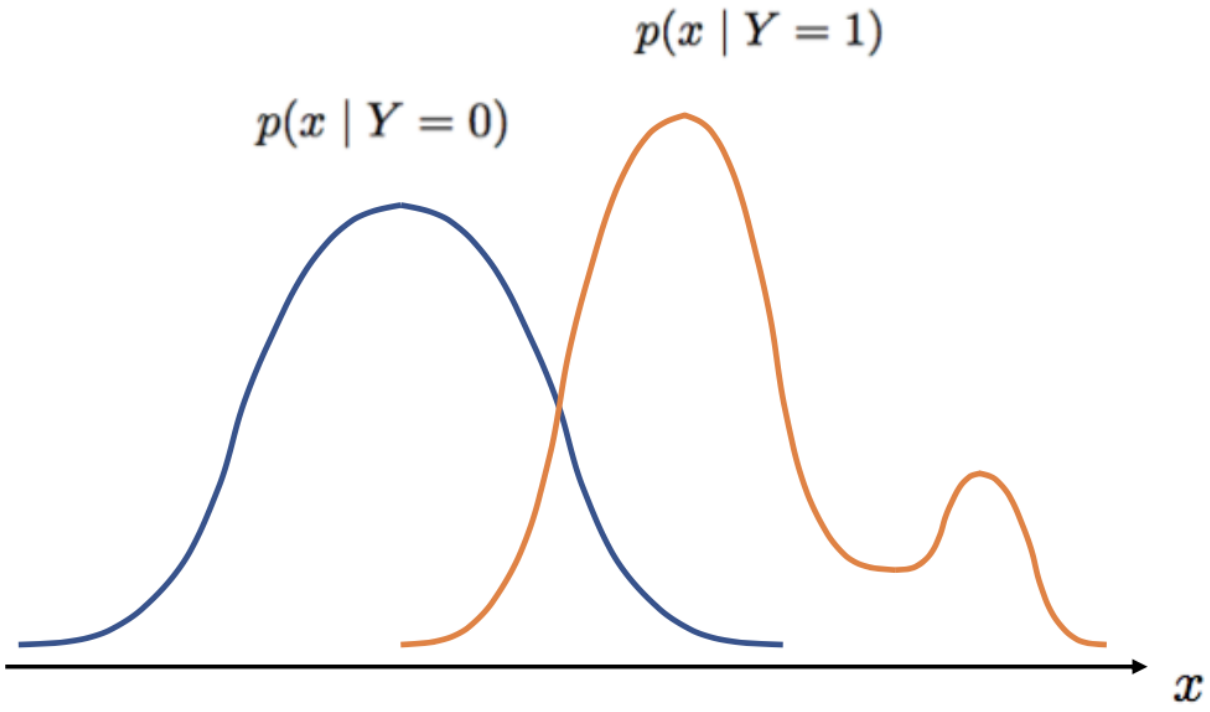


The purpose of this book is to offer a concise and rigorous introduction to PRML, while including updates on recent methods and examples of application based on the python programming language. This book does not attempt an encyclopedic treatment of PRML. A stringent selection of material is mandatory for a concise textbook, and the choice of topics made here, while dictated to a certain extent by my own experience and preferences, should reflect the core knowledge one must obtain to be proficient in PRML.

2. Basic Pattern Recognition

Class-Conditional Densities

The relative frequencies of each label as a function of predictor values are given by the *class-conditional densities* $p(x|Y = i)$, for $i = 0, 1$.



Posterior Probabilities

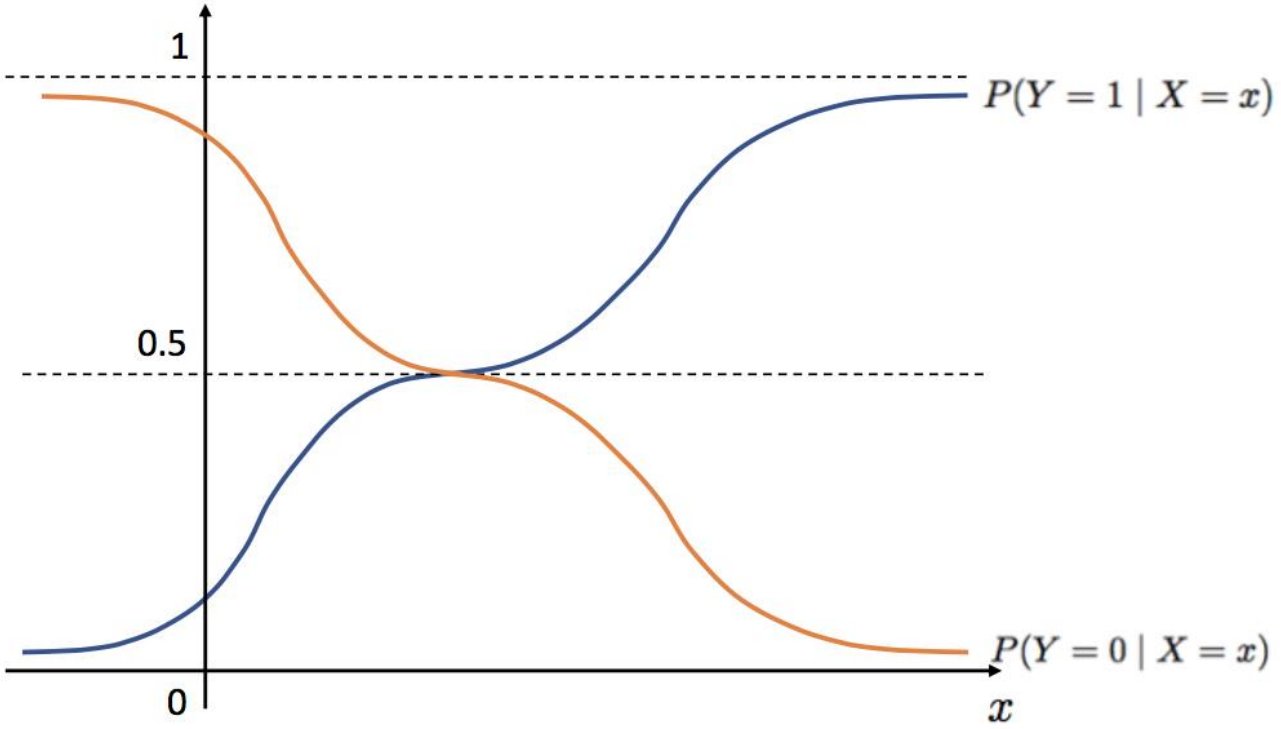
Using Bayes' theorem, we can start from the prior probabilities and class-conditional densities and find the posterior probability of $Y = i$ given that $X = x$ has been observed, for $i = 0, 1$:

$$P(Y = i|X = x) = \frac{p(x|Y = i)P(Y = i)}{p(x)}$$

$$= \frac{p(x|Y = i)P(Y = i)}{p(x|Y = 0)P(Y = 0) + p(x|Y = 1)P(Y = 1)}$$

Posterior Probabilities

Posterior probabilities are *not* probability densities (e.g., they do not integrate to 1) but are simply probabilities (in particular, they are always between 0 and 1).



Classifier and Classification Error

- Formally, a *classifier* is a (measurable) function $\psi : R^d \to \{0, 1\}$ from the feature space R^d into the binary set of labels $\{0, 1\}$. Therefore, a classifier partitions the feature space into two regions.

- The *classification error* is the probability of misclassification:

$$\epsilon[\psi] = P(\psi(X) \neq Y)$$

- This is the fundamental criterion of performance in classification. The classification error is determined by the joint distribution F_{XY}, also called the *feature-label* distribution.

Classification Error

Using the previous formulas, one can further develop the classification error as:

$$\epsilon[\psi] = \int_{x \in R^d} P(\psi(X) \neq Y | X = x) p(x)\, dx$$

$$= \int_{x \in R^d} \left(I_{\psi(x)=0}\, \eta(x) + I_{\psi(x)=1} (1 - \eta(x)) \right) p(x)\, dx$$

$$= \int_{\{x|\psi(x)=0\}} \eta(x) p(x)\, dx + \int_{\{x|\psi(x)=1\}} (1 - \eta(x)) p(x)\, dx$$

Classification Error

Now, from Bayes theorem,

$$\eta(x)p(x) = p(x \mid Y = 1)P(Y = 1)$$
$$(1 - \eta(x))p(x) = p(x \mid Y = 0)P(Y = 0)$$

Replacing these into the previous formula yields an alternative equation for the classification error:

$$\epsilon[\psi] = \int_{\{x|\psi(x)=0\}} p(x \mid Y = 1)P(Y = 1)\, dx + \int_{\{x|\psi(x)=1\}} p(x \mid Y = 0)P(Y = 0)\, dx$$

Class-Conditional Densities

We can rewrite the previous equation as:

$$\epsilon[\psi] = (1-c)\epsilon^0[\psi] + c\epsilon^1[\psi]$$

where $c = P(Y=1)$, and

$$\epsilon^0[\psi] = \int_{\{x|\psi(x)=1\}} p(x \mid Y=0)\, dx$$

$$\epsilon^1[\psi] = \int_{\{x|\psi(x)=0\}} p(x \mid Y=1)\, dx$$

are the *class-specific* error rates. Given ψ, these error rates do not depend on the prior probabilities c and $1-c$, while the overall error $\epsilon[\psi]$ clearly does.

Class-Specific Error Rates

Suppose ψ is used as a *test* to distinguish "positive" cases (class 1) from "negative" cases (class 0).

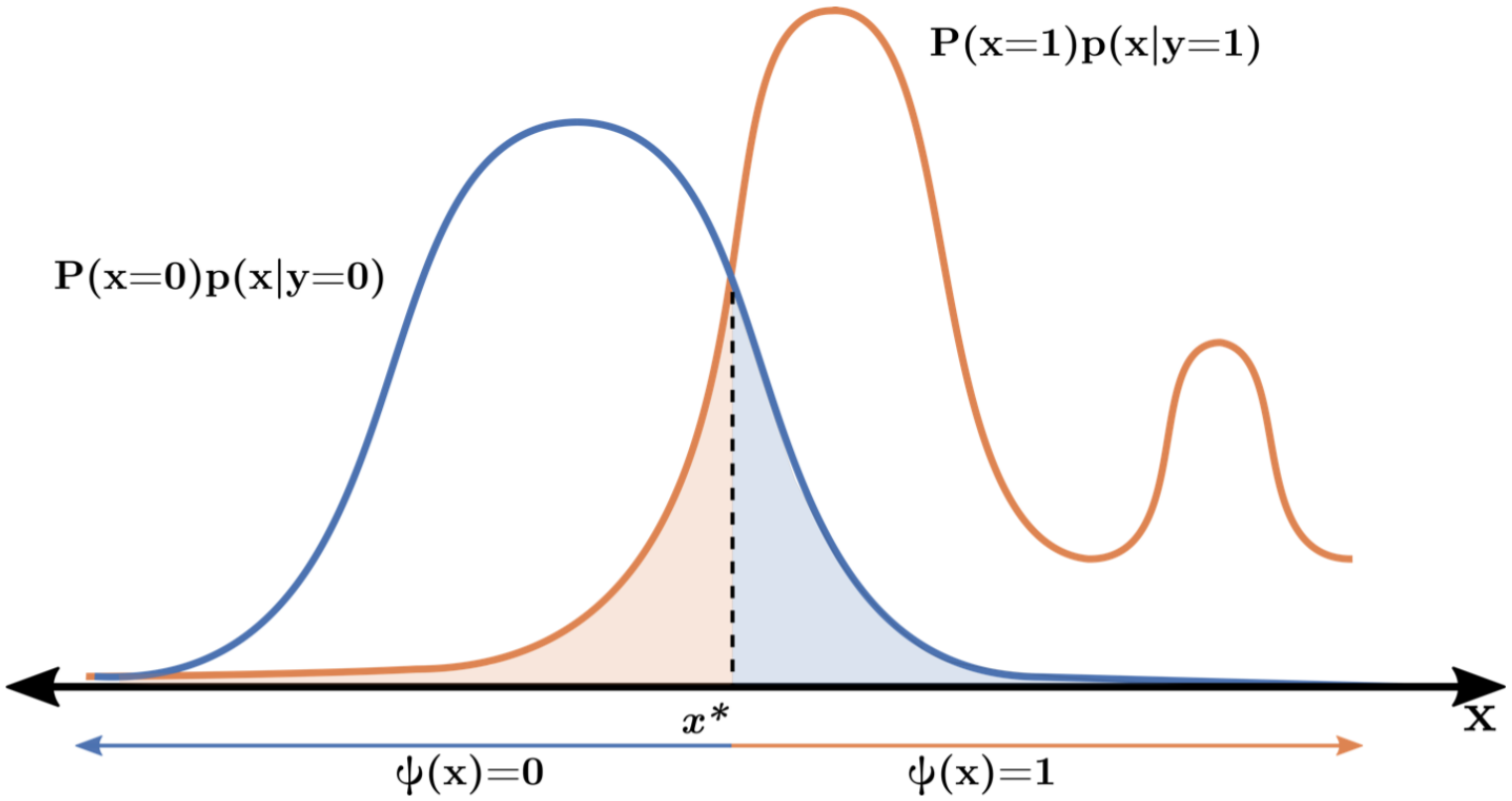
Then $\epsilon^0[\psi]$ and $\epsilon^1[\psi]$ are called the test's *false positive* and *false negative* error rates, respectively.

One also defines the test's *sensitivity* and *specificity* as

$$\text{sensitivity} = 1 - \epsilon^1[\psi] = \int_{\{x|\psi(x)=1\}} p(x \mid Y = 1)\, dx$$

$$\text{specificity} = 1 - \epsilon^0[\psi] = \int_{\{x|\psi(x)=0\}} p(x \mid Y = 0)\, dx$$

Graphical Example



The orange area is $c\epsilon^1[\psi]$, the blue area is $(1-c)\epsilon^0[\psi]$ and the error is the sum of them.

Classification Rules

- A *classification rule* is a mapping

$$\Psi_n : [R^d \times \{0,1\}]^n \to \mathcal{C}$$

where $\mathcal{C} = \{\psi \mid \psi : R^d \to \{0,1\}\}$ is a class containing all classifiers.

- This is simply saying that, given a sample $S_n \in [R^d \times \{0,1\}]^n$, the classification rule Ψ_n produces a designed classifier $\psi_n = \Psi_n(S_n) \in \mathcal{C}$.

- Note that what we have called a classification rule is really a sequence of classification rules depending on n.

Sample-Based Classification Error Rates

- Two kinds of error are of interest here. The first is the familiar classification error of the designed classifier:

$$\epsilon_n = P(\psi_n(X) \neq Y | S_n)$$

This is called the *conditional error* or *true error*.

- The conditional error is a function of the random data S_n, and therefore it is a random variable if the value of S_n is not given. The second kind of error of interest is the expected value of ϵ_n over all sample sets S_n:

$$\mu_n = E[\epsilon_n] = P(\psi_n(X) \neq Y)$$

This is called the *unconditional error* or *expected error*.

Consistent Classification Rules

- The classification rule Ψ_n is said to be (weakly) consistent if

$$\epsilon_n \to \epsilon^* \quad \text{in probability}$$

whereas it is said to be strongly consistent if

- A classification rule Ψ_n is said to be *universally* (strongly) consistent if it is (strongly) consistent for each feature-label distribution F_{XY}.

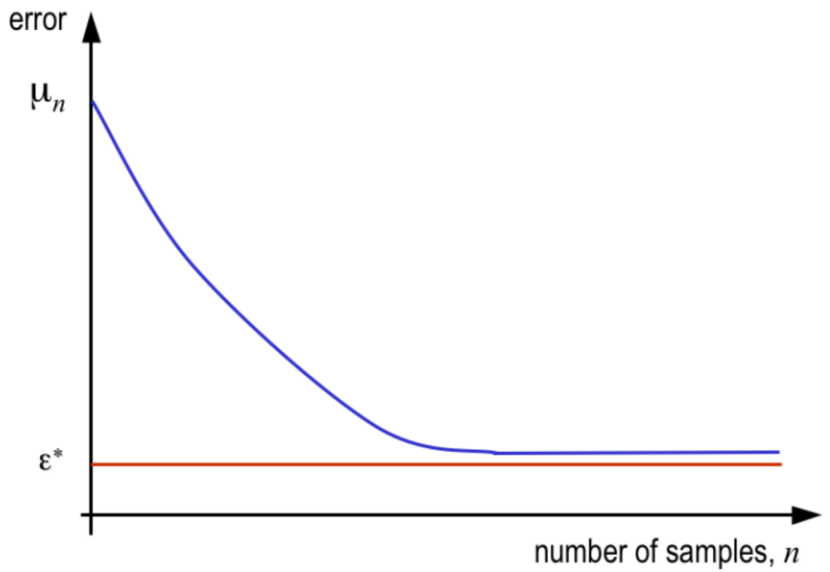
- While consistency is a property of the classification rule and the feature-label distribution, universal consistency is a property of the classification rule alone.

Consistent Classification Rules

- The following result relates (weak) consistency to the expected error.
- Theorem: Ψ_n is weakly consistent if and only if

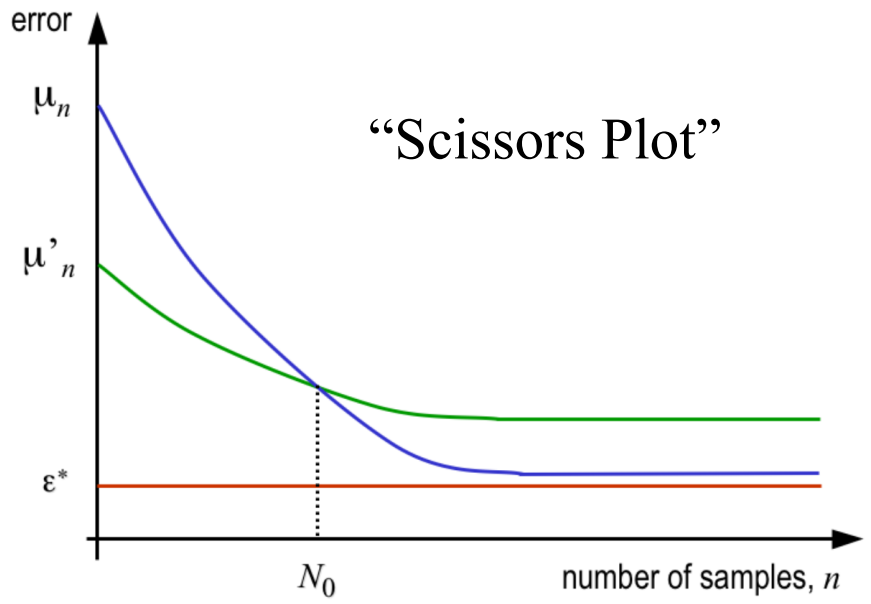
$$E[\epsilon_n] \to \epsilon^*$$

Note that this is ordinary convergence of real numbers.



Consistent Classification Rules

- A word of caution: a non-consistent classification rule may still be useful, in fact, it may be better than a consistent one, in *small-sample* scenarios.
- In the example below, the non-consistent classification rule is better than the consistent one for $n < N_0$.



"Scissors Plot"

3. Classification Error Estimation

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

- 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.

Error Estimation Variance

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

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

- 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:

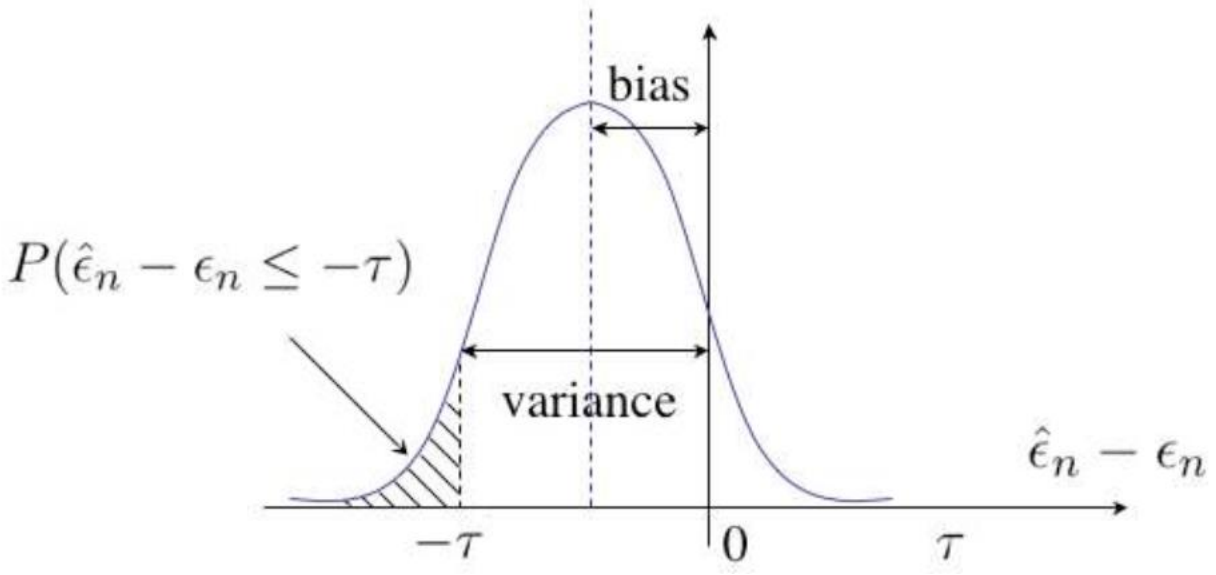
$$\text{Var}(\hat{\epsilon}_n) = E[V_{\text{int}}] + \text{Var}(E[\hat{\epsilon}_n | S_n])$$

- 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.

Deviation Distribution

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*.



Error Estimation 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*,

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

- The *root mean-square error*,

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

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

- The tail probabilities,

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

Error Estimation 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 $\tau > 0$

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

Test-Set Error 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.

Test-Set Error Estimator

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

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

Test-Set Error Estimator

- 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), $\text{Var}(\hat{\epsilon}_{n,m}) \approx \text{Var}[\epsilon_n]$

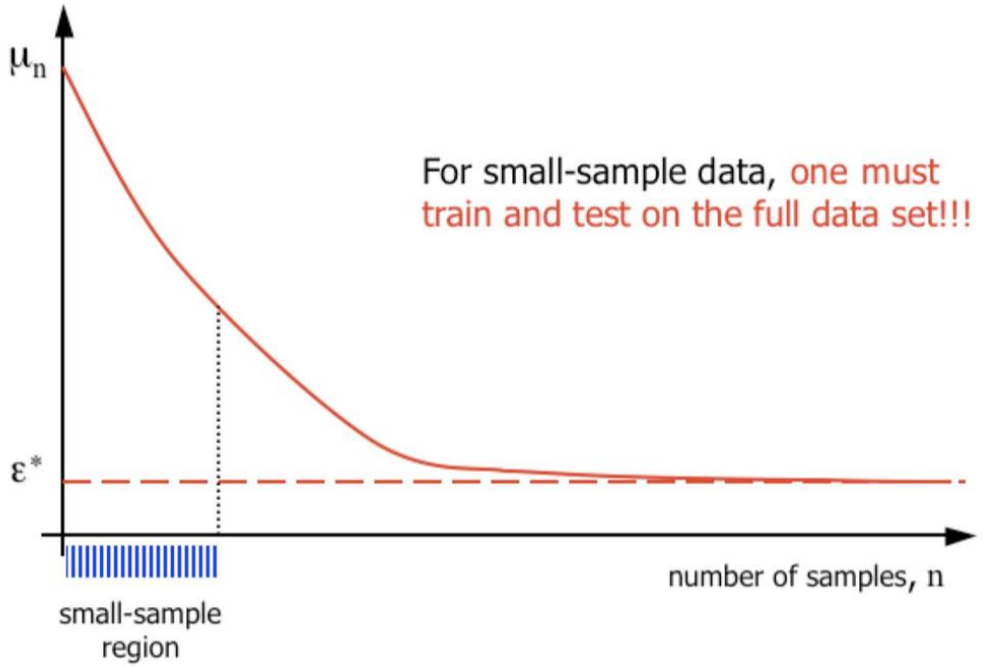
- Using Hoeffding's Inequality results in:

$$P(|\hat{\varepsilon}_{n,m} - \varepsilon_n| \geq \tau \mid S_n) \leq 2e^{-2m\tau^2}, \text{ for all } \tau > 0$$

from which it follows that the estimator is strongly consistent.

Problem with the Test-Set Error

Despite its many many nice properties, the test-set estimator has a serious drawback. In practice, there is not enough data to split between training and testing. One ends up with either insufficient training data or insufficient testing data (or both).



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.

Resubstitution

However, resubstitution can be a reliable estimator, under *simple* classification rules, in a sense that can me made precise.

If the classification rule has finite *VC dimension* (a measure of how expressive the rule is) then it can be shown, using the Vapnik-Chervonenkis Theorem, that regardless of the distribution,

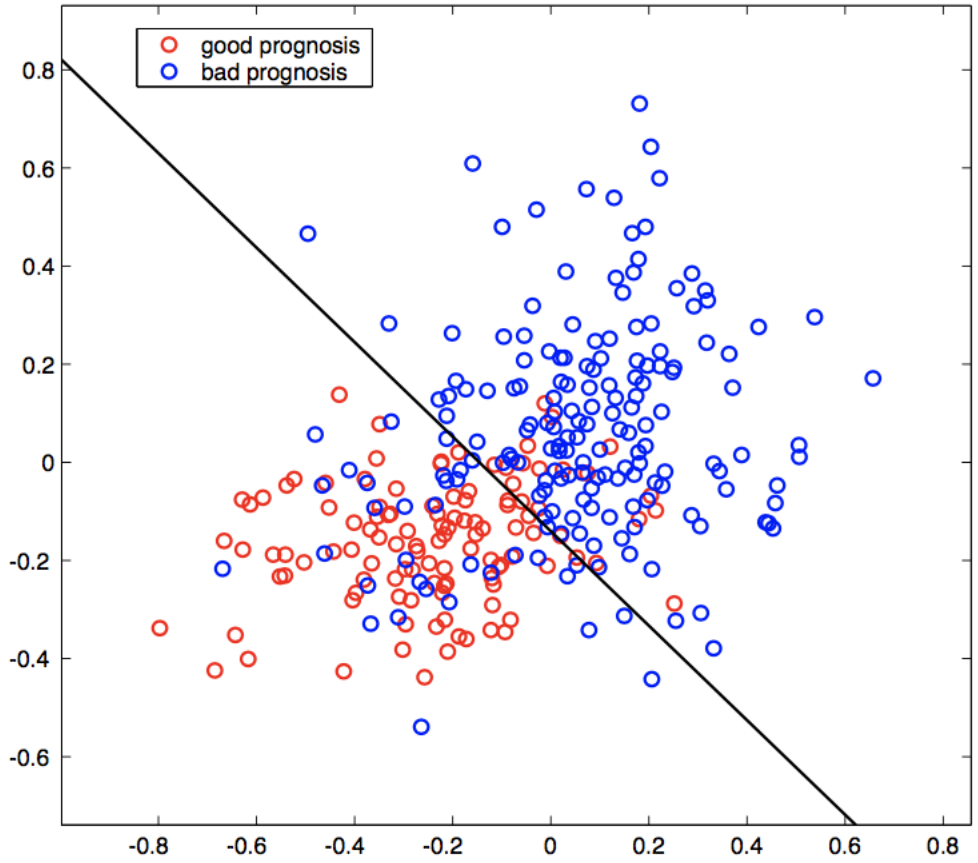
$$P\left(|\hat{\varepsilon}_n^r - \varepsilon_n| > \tau\right) \leq 8(n+1)^{Vc} e^{-n\tau^2/32}, \quad \text{for all } \tau > 0$$

from which it follows that resubstitution is strongly universally consistent.

It can also be show that $|E[\hat{\varepsilon}_n^r - \varepsilon_n]|$ is $O(\sqrt{\log(n)/n})$ as $n \to \infty$, i.e., resubstitution is asymptotically unbiased (regardless of the distribution).

Resubstitution

Example with gene-expression data



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

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

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:

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

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

Cross-Validation

- 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 Cross-Validation

- 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

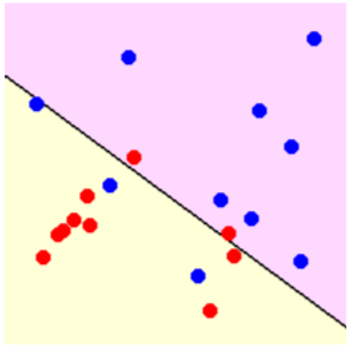
$$\psi_{n-n/k}^{i} = \Psi_{n-n/k}(\cdot; S_n \setminus S_{(i)}) \quad i = 1 \ldots, k$$

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

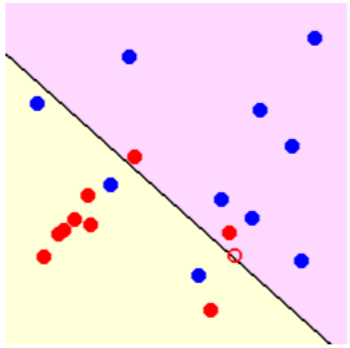
- It also makes $\hat{\epsilon}_n^{cv(k)}$ an approximation of the *expected* classification error $E[\epsilon_{n-n/k}]$ rather than ϵ_n or $\epsilon_{n-n/k}$.

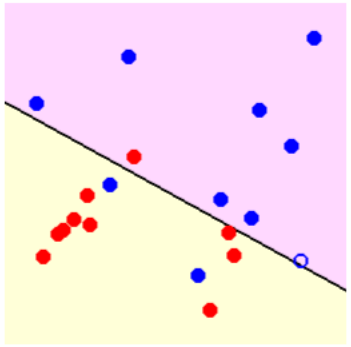
Example: Surrogate LDA Classifiers

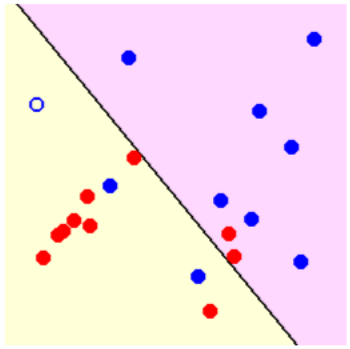
Original LDA classifier



A few surrogate LDA classifiers

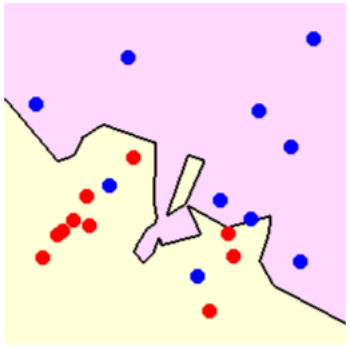




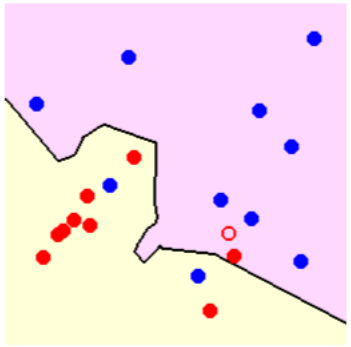


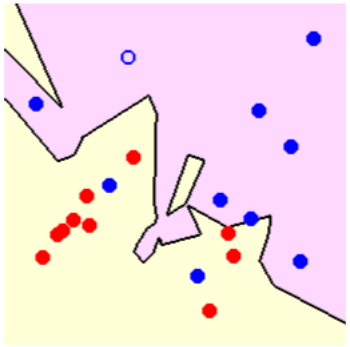
Example: Surrogate 3NN Classifiers

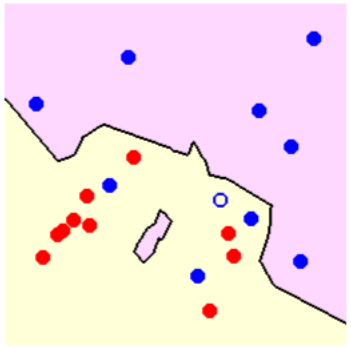
Original 3NN classifier



A few surrogate 3NN classifiers

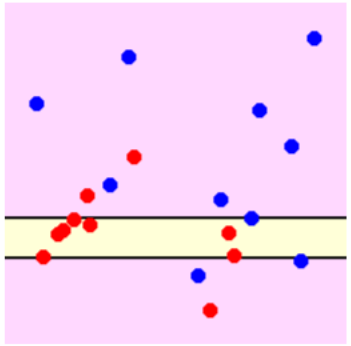




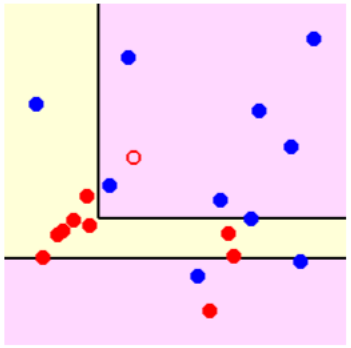


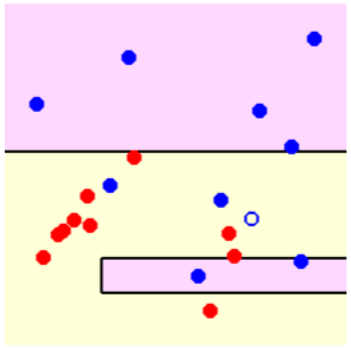
Example: Surrogate CART Classifiers

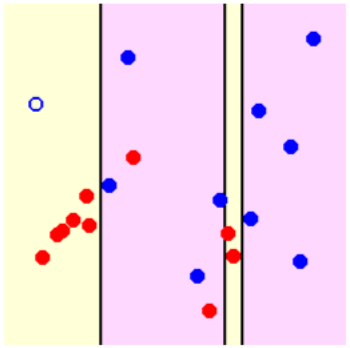
Original CART classifier



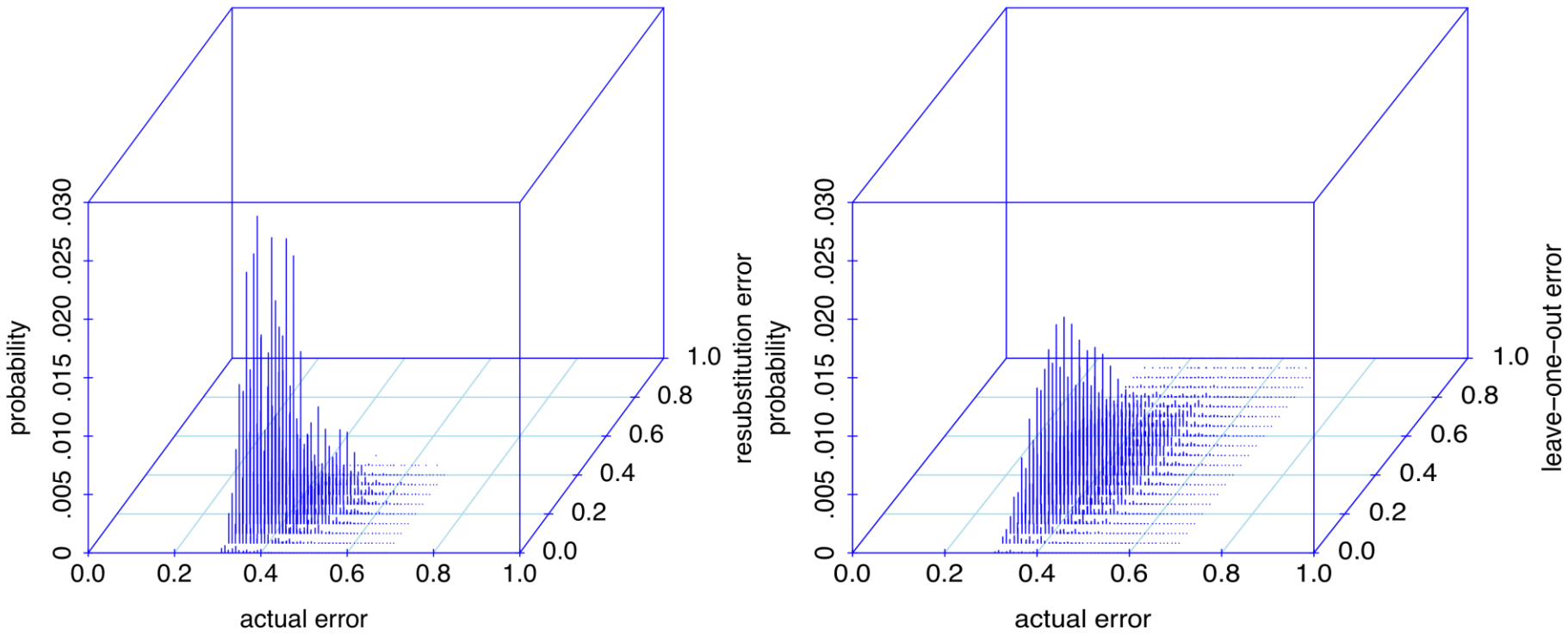
A few surrogate CART classifiers







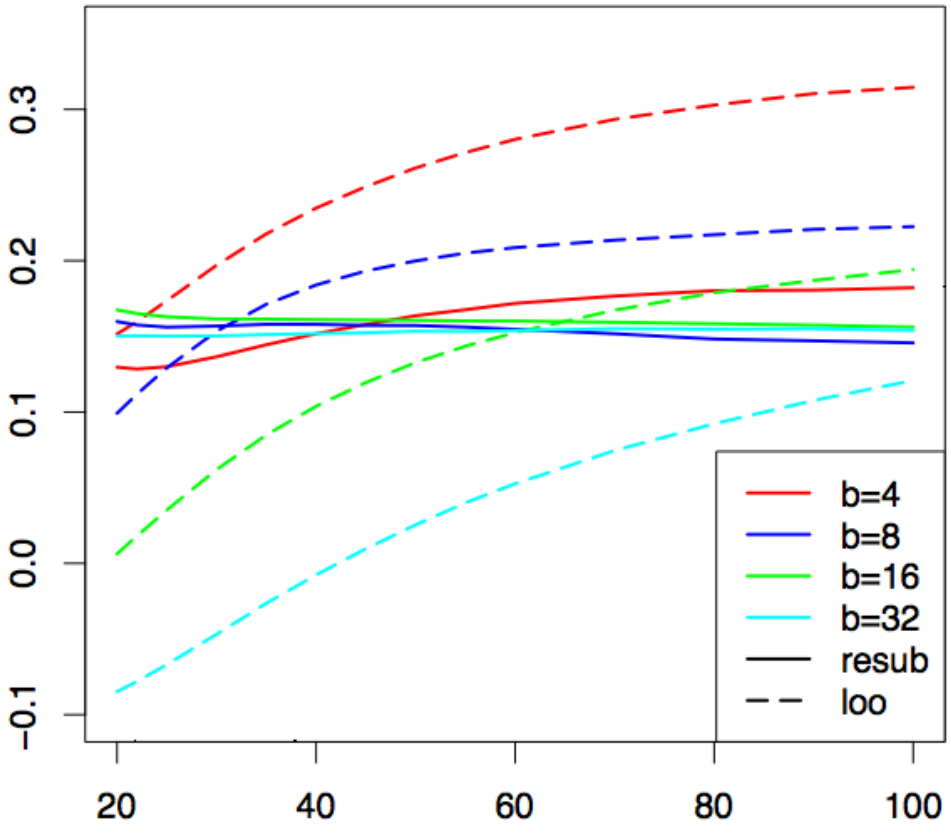
Variability of Cross-Validation



Exact joint distribution between actual error and resubstitution and leave-one-out error estimators, discrete histogram rule with n=20 samples and b=8 cells.

Negative Correlation

- Cross-validation (leave-one-out) can be shown to display negative correlation with true error.



correlation vs. sample size, Bayes err=10%

U.M. **Braga-Neto** and E.R. Dougherty, Exact Correlation between Actual and Estimated Errors in Discrete Classification, *Pattern Recognition Letters*, Vol. 31, No. 5, April 2010, pp. 407-412

Bootstrap Error Estimator

- 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 Error Estimator

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

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

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.

Bootstrap Error Estimator

- 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.

Bolstered Error Estimation

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 Error Estimation

- 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 Error Estimation

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

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

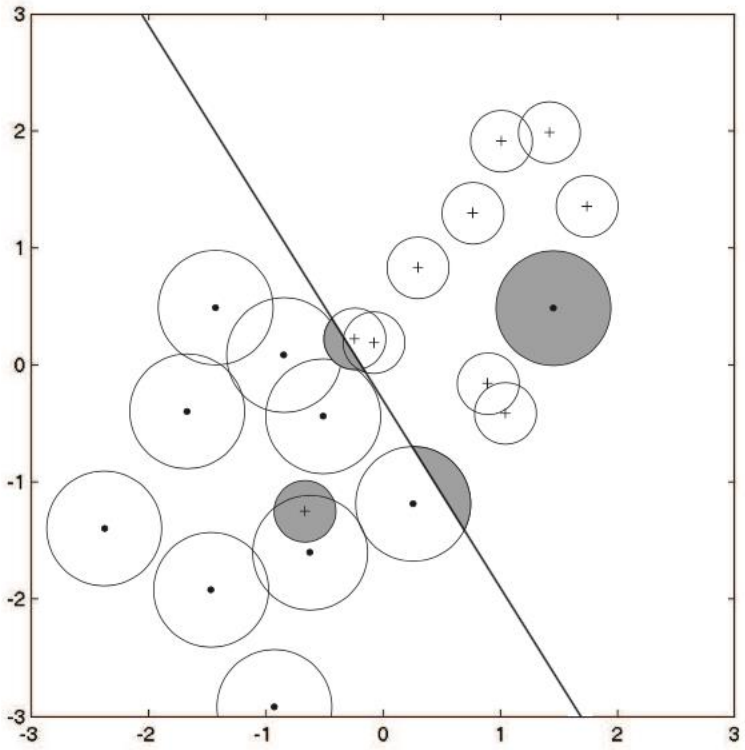
- The expectation can be written out as follows

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

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

Bolstered Error Estimation

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



Error Estimation 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

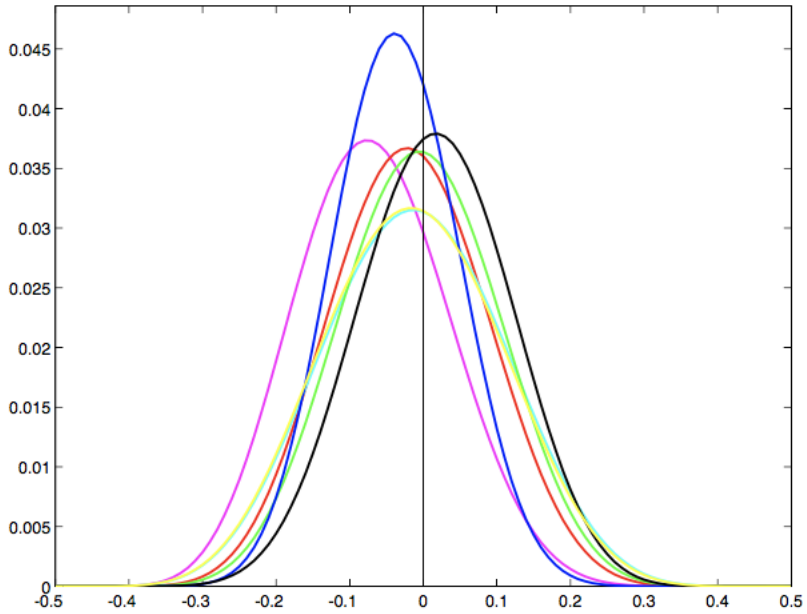
Simulation Example

- 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 $\hat{\epsilon}_n - \epsilon_n$.

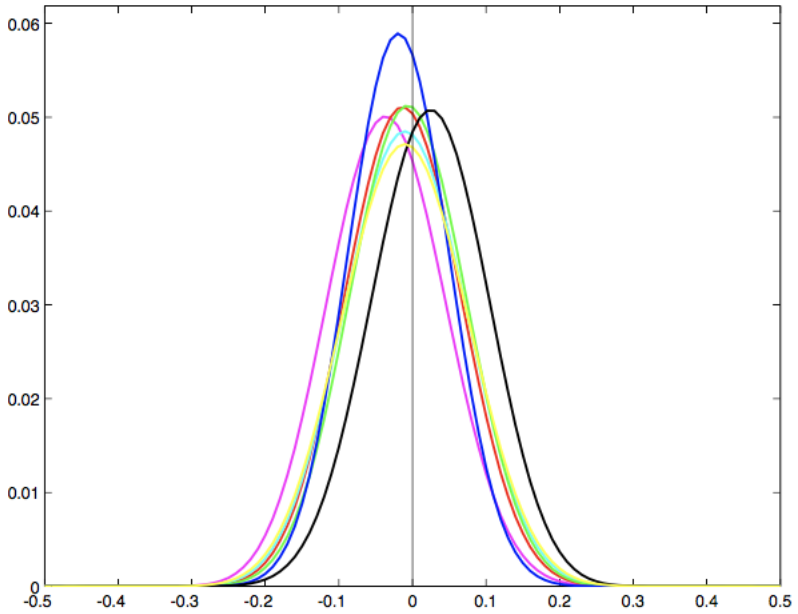
Deviation Distributions - LDA

resub loo cv10r b632 bresub sresub bloo

$n = 20$



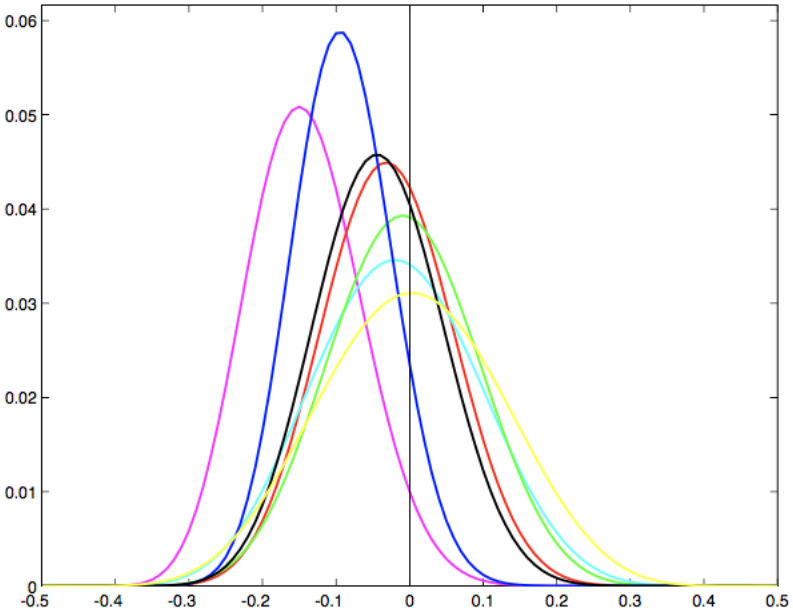
$n = 40$



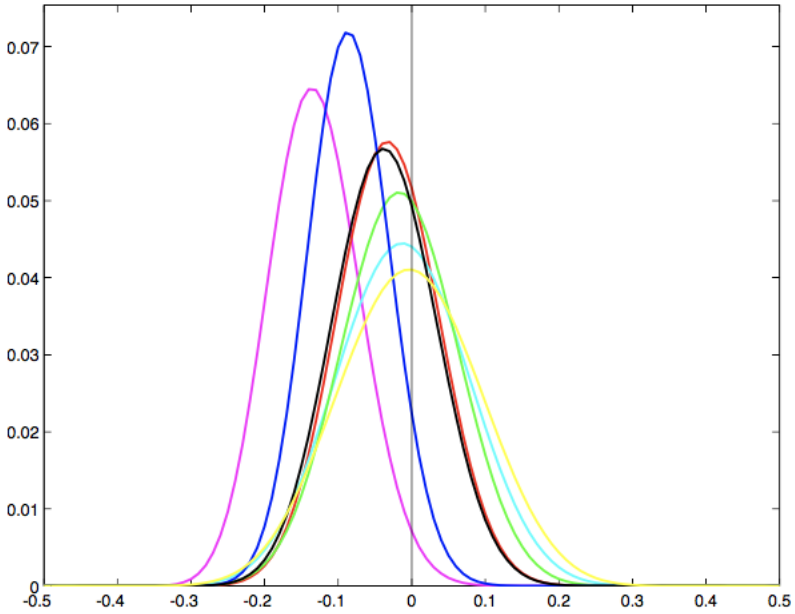
Deviation Distributions – 3NN

resub loo cv10r b632 bresub sresub bloo

$n = 20$



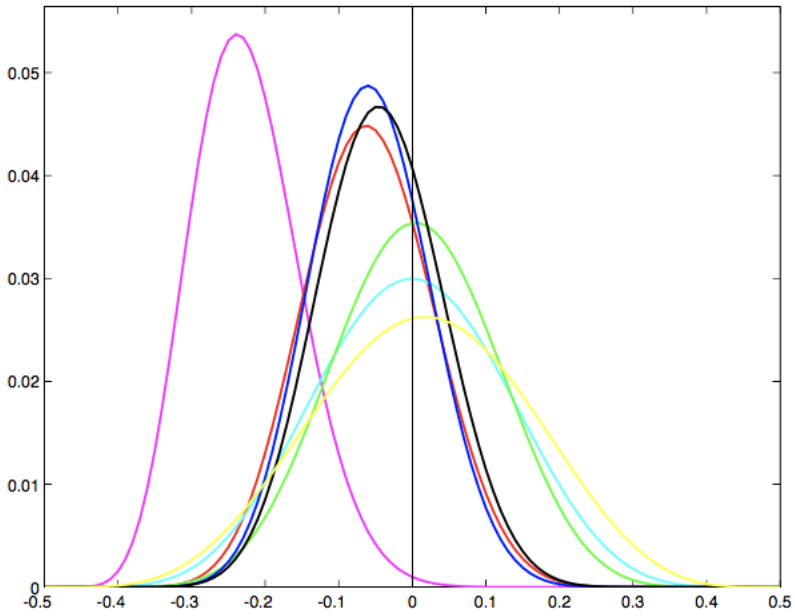
$n = 40$



Deviation Distributions - CART

resub loo cv10r b632 bresub sresub bloo

$n = 20$



$n = 40$