UPTEC X 05 022
MAY 2005
ISSN 1401-2138

MIKAEL WALLMAN

Performance estimation in small sample parameter and classifier selection

Master’s degree project

UPPSALA UNIVERSITET

Molecular Biotechnology Programme

Uppsala University School of Engineering

UPTEC X 05 022	**Date of issue 2005-05**

Author

Mikael Wallman

Title (English)

Performance estimation in small sample parameter and classifier selection

Title (Swedish)

Abstract

In this work it was shown that improper use of performance estimation on classifiers optimized by cross validation induces a positive bias in the error estimate, especially for small sample number data sets typically used in biological applications. The mechanism behind the bias was investigated, and a remedy was suggested in the from of a double cross-validation loop. However, optimization procedures in conjunction with requirements for unbiased error estimates give less reliable performance estimation. This problem was addressed with Bayesian inference and consequences of the decreased reliability were investigated.

Keywords

performance estimation, bias, cross-validation, Bayesian inference.

Supervisors

Mats Gustafsson
Department of engineering sciences, Uppsala University

Scientific reviewer

Jan Komorowski
LCB, Uppsala University

Project name	Sponsors
Language **English**	Security
ISSN 1401-2138	Classification
Supplementary bibliographical information	Pages **25**

Biology Education Centre Biomedical Center Husargatan 3 Uppsala
Box 592 S-75124 Uppsala Tel +46 (0)18 4710000 Fax +46 (0)18 555217

Performance estimation in small sample parameter and classifier selection

Mikael Wallman

Sammanfattning

Lärande system för mönsterigenkänning har på senare år fått en rad nya tillämpningar inom bioinformatik och datorbaserad biologi. Den bakomliggande principen går ut på att vi genom att träna en algoritm kan lära den att klassificera nya observationer. Ett centralt begrepp inom lärande system är felfrekvensestimering, dvs metoder att förutsäga hur ofta vi kommer att göra fel för ett givet problem och en given träningsprocedur. Det har genom åren föreslagits en rad olika felestimeringsmetoder. Så fort vi har en metod för felestimering öppnar sig emellertid möjligheten att optimera. Detta sker genom att helt enkelt utvärdera flera olika algoritmer, tränade med samma mönster, och sedan välja den som förefaller bäst.

Optimering med hjälp av vissa felestimeringsmetoder kommer dock att leda till felaktiga felfrekvensestimat. I det här arbetet har fenomenet undersökts, och dess existens bekräftats med hjälp av datorsimuleringar. Som bot föreslås även en metod för att skaffa sig korrekta estimat vid optimering. Denna metod leder dock till ökad osäkerhet om den sanna felfrekvensen. Konsekvenserna av denna ökade osäkerhet har undersökts med statistiska metoder. Slutsatsen är att det under vissa förutsättningar kan vara bättre att låta bli att optimera.

Examensarbete 20p i Molekylär bioteknikprogrammet

Uppsala universitet maj 2005

Contents

1 Introduction

The aim of algorithms for supervised learning is to make a computer able to learn from examples, thus also making it able to generalize when presented with new input. When this generalization consists of assigning a class to a presented pattern, the algorithm is often termed classifier. Typical applications for this type of algorithms are found in fields as diverse as speech recognition, radar image analysis and weather forecasting. Recent years have also seen a growing number of applications in the subject of computational biology. There is a large number of algorithms available for achieving this type of tasks, ranging from relative simplicity to extreme complexity [1], [2].

Once we have trained our algorithm by presenting it with data and telling it how to respond, we are usually interested in knowing how well we should expect it to perform when presented with new data. A common measure of performance is the so-called error rate. Since we usually have no way of analytically calculating the error rate, we must rely on estimation techniques. A widely used method for error rate estimation is cross validation (CV).

With the ability of estimating the error rate comes the possibility of optimization. If we can estimate the error rate of our algorithm with one set of parameters, we can just as well produce an estimate for other sets of parameters. We could then simply select the set of parameters yielding the lowest error rate estimate, and use it to design our classifier. This way we both get the most promising classification algorithm and an estimate of its error rate - a perfect solution, it would seem.

However, a commonly overlooked issue is that the optimization procedure described above will induce an optimistic bias on the error rate estimates produced [3], [4]. This is basically due to the fact that parameter tuning increases the risk of overfitting the classifier to the data. In bioinformatical litterature, often both biased and unbiased error estimates are reported [3], and in certain cases only the biased ones [5]. In this work, the existence of this phenomenon was assessed by computer simulations and the underlying mechanisms were further investigated and explained.

A way of obtaining an unbiased error rate estimate when optimizing parameter selection with CV is to test the resulting classifier on previously unused data. In this work, this approach is represented by a double cross validation loop, where an optimal parameter set is selected in the inner loop, and an unbiased error rate estimate is obtained in the outer loop. It should be noted

that the error rate estimate produced by the outer loop will be an estimate of the performance for the whole classifier selection procedure and the resulting classifier, and not just for a classifier with a certain choice of parameters.

It is commonly known that the smaller number of samples we have in our test data set, the less reliable our error rate estimate will be. Since the procedure described above does not allow for the same data to be used in both optimization and testing, test data is "consumed" by optimization (or, if you will, optimization data is "consumed" by testing). This leaves us with something not unlike Heissenberg's principle of uncertainty - if we use all data for optimization, we have no idea about the performance of the resulting classifier, but if we use all data for testing, the optimization process will be a game of random guessing. Apparently we find ourselves in a trade-off situation.

Setting this trade-off problem aside for a moment, an intermediate issue to consider is that CV really provide no more than a sample from an error rate distribution, of which the mean value represents the true error rate [1]. The true error rate here refers to the error rate that would be observed for a classifier built on all currently available data and tested on a very large amount of new, independent, data points from the same underlying distribution. Iterating the CV-process with new data from the same underlying distributions will, as shown below, eventually make us able to form this error rate distribution and calculate the average. However, under real life circumstances only one data set is usually available, which makes iteration of the process impossible.

Thus, we need some way of coping with the uncertainly inherent in the error estimation process. One way of achieving this is by use of Bayesian inference. This enables us to calculate a posterior probability density function (pdf), describing our uncertainty about the true error rate [1]. This way we gain the ability to mathematically treat not only the error rate itself, but also the amount of knowledge we have about it.

Now, remembering the above discussion about the trade-off problem, we must realize that the use Bayesian inference will of course not in any way prevent the loss of data from increasing our uncertainty, but merely make us able to handle it mathematically. Hence, the trade-off problem remains unsolved.

To summarize, our uncertainty is increased by loss of data when trying to optimize our classifier. A way to measure this uncertainty is by using Bayesian inference to calculate a pdf of the true error rate. Here new questions arise:

What problems does this increased uncertainty cause? In particular, it would be interesting to know if optimization is worth the effort at all. Do situations exist where it is preferable to just settle for a reasonably robust classifier over going though a resource-consuming optimization process, according to some performance measure?

In this work, these two questions have been addressed by means of simulations, aimed at comparing joint parameter and algorithm optimization to a pre-defined choice of classifier on several different performance measures and data distributions.

2 Background

2.1 Supervised Learning

The general process of supervised learning and classification can be described as follows: We have some kind of measurement, for instance data from a microarray measurement of mRNA concentrations. Associated with these measurements are some type of classes, for example the two classes sick and healthy. Now we put in the measurements and their corresponding classes into the learning algorithm, thus making it adapt to the data. This constitutes the training (or design) procedure. If we now present the algorithm with new measurements, lacking corresponding classes, the idea is that it should be more likely to put put the correct corresponding classes than it was before the training procedure.

Based on the above, the classification procedure can be seen as a function, transforming the data from n to 1 dimension, where n is the number of parameters taken into account in each measurement. For instance, if our measurements consists of 1000 gene expression levels from a patient, and our algorithm is trained to decide whether its input corresponds to a sick or healthy person, the classification process constitutes a transform of a 1000-dimensional vector to a scalar. Since, in reality, some patterns of gene expression levels actually do correspond to one class, and others to another class, the training process can be viewed as trying to adapt the parameters of the transforming function to make it as similar as possible to the real underlying function. This function will be denoted the target function.

2.2 Error Estimation

2.2.1 Hold-out Tests

One of the most common measures of a classifiers performance is its error rate or misclassification rate, i.e. the number of misclassifications divided by the number of test examples. Since we seldom know the target function, the possibility of analytically determining the error rate is very rare. The true error rate is the expected probability of misclassification on a randomly selected test set. It could hypothetically be calculated by testing the classifier on an infinitely large test set generated by the same target function as the training set.

A naive approach to error rate estimation would be to test the classifier on the same data with which is was trained. This type of error estimate is called the apparent error rate. It will generally suffer from substantial optimistic bias, since many algorithms can learn a training set perfectly or near perfectly, and thus yield an apparent error of zero or near zero.

A far more used method of error rate estimation is known as hold-out error estimates. A holdout estimate is derived by testing the classifier on a holdout data set, i.e. a set which has not been used for training the algorithm. If there is no overlap between the training set and the holdout set, the resulting error estimate is called a off-training set estimate.

2.2.2 Cross-validation

In biological applications of pattern recognition, the available data sets typically contain a small number of samples. With a very limited set of labeled samples, the need for using the available information efficiently increases. A technique often employed under these circumstances is cross-validation (CV). In CV, the data is partitioned into k equally sized subsets. Each subset is then used as a test set for a classifier built on the remaining $k-1$ subsets. Each test will render an error rate, and the average of these k error rates is the approximation of the true error rate. If k is equal to the total number of available samples, the procedure is often referred to as leave-one-out cross-validation (LOOCV). Since k classifiers must be designed in order to perform a single CV, the method is computationally intense. On the other hand, the error rate estimate produced is approximately unbiased and have

a decreased variance compared to that from a simple holdout test.

2.3 The No Free Lunch Theorem

The No Free Lunch theorem (NFL) states that the average performances of any two classifiers over all possible classification problems will be the same [6], [7]. This means that our classification algorithm of choice will be outperformed in exactly as many cases as it has the best performance, compared to any other algorithm (even random guessing), as long as we lack information about the nature of the classification task. Thus, if we want to compare the performance of several classifiers, we must do it with respect to a particular problem or set of problems.

3 Materials and Methods

3.1 Computer Programming

Development and simulations were performed in MatLab (MathWorks Inc., Natick, Mass; USA), using the PRTools toolbox [8].

3.2 Data Generation

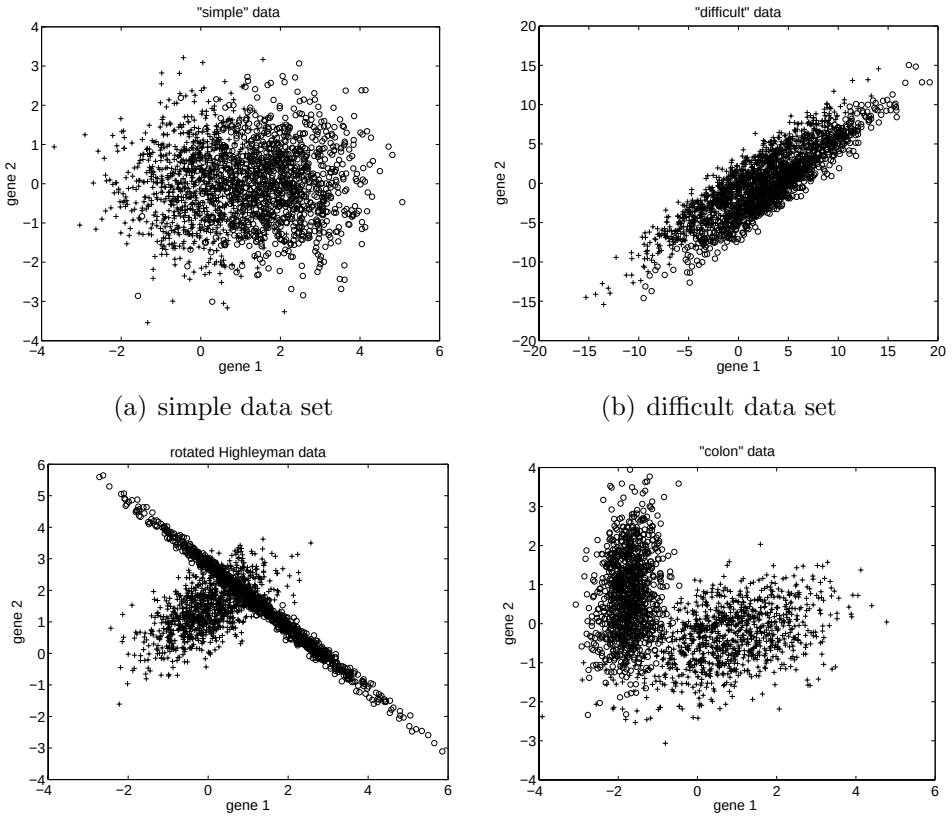
Test data from four different underlying distributions was generated (see Fig. 1). All underlying distributions consisted of two partially overlapping gaussian distributions, one per class. All generated data sets were 2-dimensional. Three of the four underlying distributions, the "simple", the "difficult" and the Highleyman distribution, were obtained from the MatLab PRTools toolbox [8] (see Fig. 1(a), 1(b) and 1(c)). The "colon" data set was derived using a generative model based on two normally distributed gene expression patterns (see Fig. 1(d)). The mean values and covariance matrices were extracted from a set of real microarray gene expression examples [11].

For the double cross validation loop, 1000 data sets from the "simple" distribution were generated. Each data set contained 25 examples from each of

two classes. In addition to this, a large test set made up of 5000 examples from each of the two classes was generated for validation purposes.

For the algorithm comparing an optimization procedure to a single classifier, 4000 data sets were generated (1000 from each underlying distribution). Each data set consisted of 180 exampels, 90 from each of the two classes. In addition, four large validation sets were also generated, one from each of the four different underlying distributions.

1000 samples per class from the four distributions used in this work are shown in Fig. 1.



(a) simple data set

(b) difficult data set

(c) rotated Highleyman data set

(d) colon data set

Fig. 1: Scatter plots (1000 points/class) of the four different two-class distributions used in the simulations.

3.3 Classifiers

Four classifiers were used in this work: the k-nearest neighbour classifier [1], the Parzen density classifier [9], a tree-based classifier and Fishers linear discriminant [1]. The implementations of the classifiers used are part of the PRTools toolbox.

3.4 Double Cross Validation Loop

As mentioned in Section 1, a double CV-loop was implemented to assess the bias in CV optimization. Each one of the 1000 data sets from the "simple" distribution (see Fig. 1), consisting of 25 examples from two partially overlapping Gaussian distributions (50 examples in total, one class per distribution), was partitioned into 10 subsets of roughly equal size. At each iteration, 9 of these subsets were used to form an external design set and the remaining one to form the external validation set.

The external design set was then used to calculate a LOOCV error estimate for all possible choices of k for a kNN classifier. In principle, any classifier could have been used here, but since a method for very time efficient CV is applicable to kNN [10], this classifier was an attractive choice for the experiment.

The choice of k which produced the lowest internal (LOOCV) error estimate was used to design a kNN classifier from the entire external design set. In the event of a draw between two or more k-values, the lowest one was chosen. The resulting classifier was tested on the external validation set, to obtain an external error estimate, and on a large (5000 examples from each distribution) independent validation set to produce a 'true' error rate. This procedure was repeated for all choices of external design- and validation sets, resulting in 10 internal and 10 external error estimates for each of the 1000 data sets. The double loop CV procedure is illustrated in principle by Fig. 2 and 3.

It should be noted that for purposes of clarity and convenience, the procedure in Fig. 2 and 3 only contains a 3-fold external CV (instead of 10-fold), a 3-fold internal CV (instead of LOOCV) and a choice between 3 classifiers (instead of all possible choices of k for kNN).

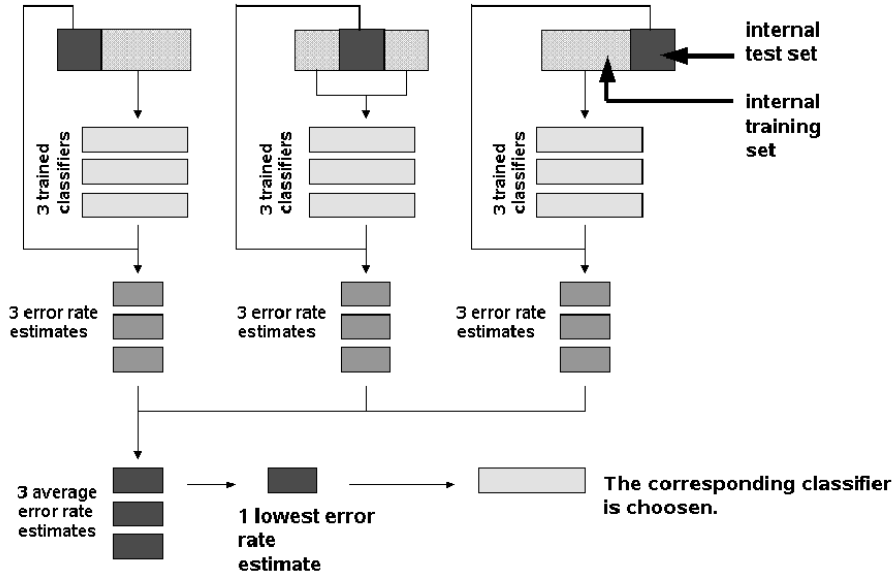


Fig. 2: (b) The internal CV-loop, illustrated by a 3-fold CV procedure selecting between 3 different classifiers.

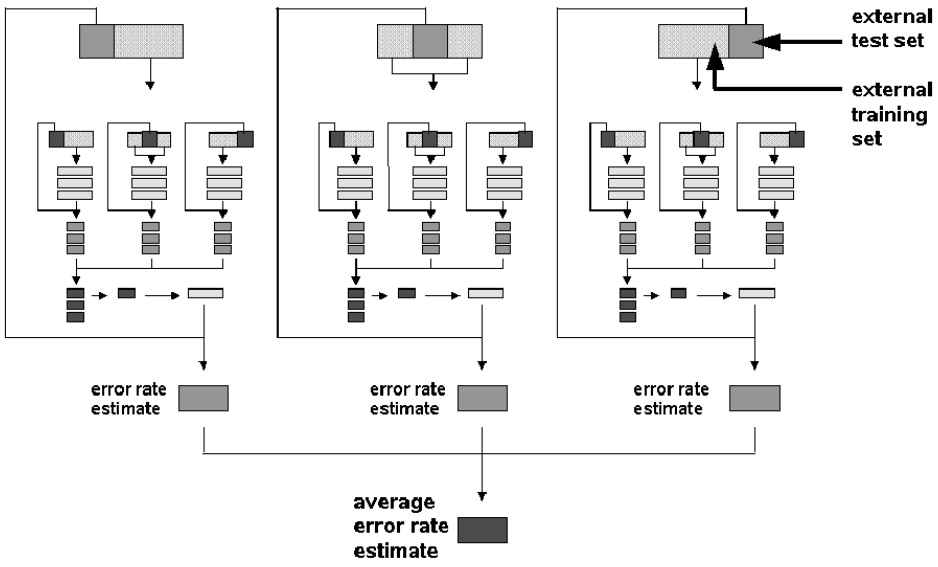


Fig. 3: The entire double CV procedure, illustrated by a 3-fold external CV loop, applied to the 3-fold internal CV loop selecting between 3 different classifiers.

3.5 Bayesian Calculation of pdf for the True Error Rate

Before moving on to the algorithm for comparing a pre-defined choice of classfier to a CV-optimization, the calculation of the Bayesian pdf:s need to be explained. Bayes' theorem is often formulated as:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{1}$$

This theorem allows us to describe the posterior probability $p(A|B)$ in terms of a prior probablilty $P(A)$, a conditional probability $P(B|A)$ and a normalization factor $P(B)$. With the aid of (1), it is possible to calculate a pdf for the true error rate [1].

Given the holdout error (HO) and the size of the training set, we can obtain a confidence interval of the true error rate according to:

$$p(k|e_T, n) = \binom{n}{k} e_T^k (1 - e_T)^{n-k} \tag{2}$$

This expression gives us the probability for k out of n samples being misclassified, given that the true error rate is e_T. Using Bayes' theorem we can then obtain the probability for the true error rate, given the number of samples misclassified:

$$p(e_T|k, n) = \frac{p(k|e_T, n)p(e_T, n)}{\int p(k|e_T, n)p(e_T, n)de_T} \tag{3}$$

Now, given that $p(k|e_T, n)$ is binomially distributed and assuming that $p(e_T, n)$ does not vary with e_T, we obtain:

$$p(e_T|k, n) = \frac{e_T^k (1 - e_T)^{n-k}}{\int e_T^k (1 - e_T)^{n-k} de_T} \tag{4}$$

This equation gives the pdf for the true error rate e_T, given information about how many examples used for testing, and how many errors made on a holdout test. It describes our uncertainty about the value of true error rate.

3.6 Reversed CV

The reversed CV (rCV) is best defined by how it differs from ordinary CV. The difference between rCV and ordinary CV is that the training- and test sets are interchanged, that is, in rCV the classifier was trained on one subset and tested on the remaining $k - 1$ subsets.

3.7 Computer Simulations for Comparing CV-optimization to a Pre-Defined Choice of Classifier

As stated in Section 1, in addition to the double CV-loop, a procedure for comparing a pre-defined choice of classifier to a CV optimization was implemented. Again, as in the double CV loop, it should be noted that the choice of available classifiers in the CV optimization, as well as the pre-defined choice of classifier, is unimportant in principle. Here, the following two algorithms were compared:

Selection between 3NN, tree classifier, Parzen density classifier and Fishers linear discriminant, based on 5-fold rCV error estimates. (a1)

3NN (a2)

The resulting trained classifier for each algorithm was tested, using holdout sets of different sizes. Each algorithm initially had the same 180 examples at their disposal. For optimizing the selection of classifier in algorithm (a1), a data set of 120 examples was used, from which 30 examples were subsequently drawn at random to train the classifier chosen by each selection procedure. The resulting classifier was then tested on a test set consisting of the remaining 60 examples. Algorithm (a2) was trained using 30 examples and tested with the remaining 150 examples.

Four performance measures were considered:

Upper boundary of the 95% confidence interval of $p(e_T|k, n)$ (ub)

Probability mass above 50% of $p(e_T|k, n)$ (50%)

Probability mass of $e_T^2 \cdot p(e_T|k,n)$ (q2)

The HO-error estimate (err)

All error measures except (err) were calculated according to Eq. (3), using HO-tests mentioned above. The (ub)-measure is, with a certainty of 95%, the highest possible error rate we can expect to encounter. The (50%)-measure tells us with what probability the true error rate lies above 50%, that is how likely it is that our classifier will perform worse than random guessing. The (q2)-measure consists of the entire probability mass, weighted by a exponentially increasing cost factor, penalizing pdf:s with large probability for high true error rates. The (err)-measure, included as a form of reference, is the resulting error rate estimate from a simple holdout test.

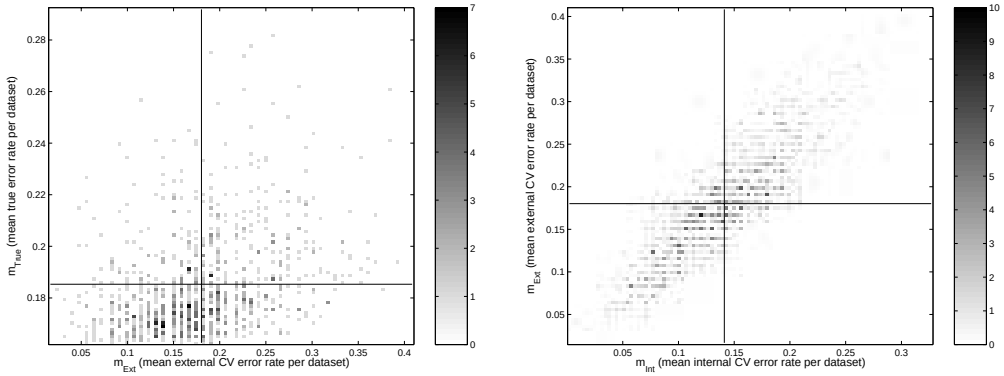
For each performance measure, the value is inversely proportional to how well the algorithm does, since a high value means bad performance according to all performance measures considered. For each performance measure and each iteration, the performances of the two algorithms were compared. The one which performed better was appointed winner, thus rendering four wins per iteration, one for each performance measure. The total number of wins for each of the two algorithms was then counted, rendering four values (scores) per algorithm. The process was tested on the four different distributions described in Section. 3.2 (see Fig. 1), with 1000 iterations for each distribution.

4 Results

4.1 Computer Simulations for Identification of CV Bias

Experiments for assessing the bias in CV optimization were carried out as described in Section 3.4. The correlations between the three mean error rates (internal, external and 'true') for each of the 1000 data sets are shown in Fig. 4. Each pair of different mean error rates rendered one point (x_i, y_i) = (mean error rate x_i, mean error rate y_i) for each data set. Figures were constructed by dividing the range between the largest and smallest value on each axis into 100 equally sized sections, thus forming a two dimensional grid. For each square in the grid, the number of points was counted. The resulting

matrices were finally visualized by giving each square a color corresponding to the number of points it contained, spanning from white (no points) to black (many points). Fig. 4 show correlation between internal and 'true' mean error rate, correlation between external and 'true' mean error rate and correlation between internal and external mean error rate respectively. The center of each distribution of mean error rates is marked by a black line. As can be seen from Fig. 4, and from Table 1, the averages of the internal mean error rates and the external mean error rates differ significantly, while the difference between the averages of the external and the 'true' mean error rates is much smaller.



(a) external vs true

(b) internal vs external

(c) internal vs true

Fig. 4: Correlation between different error rate measures. Centers of distributions are marked with black lines.

The standard deviations in Table 1 were calculated as follows: The double CV-loop applied to each of the 1000 data sets resulted in 3×10 errors (10 internal, 10 external and 10 true). The 3 different means of these 3×10

errors were calculated for each data set, resulting in 3×1000 mean error rates. These means are denoted m_{int}, m_{ext}, and m_{true}, where m_{int} is the 1000 mean values of the 10 internal errors per data set, m_{ext} is the 1000 mean values of the 10 external errors per data set and m_{true} is the 1000 mean values of the 10 true errors per data set. The standard deviations of these mean values are denoted std_{int}, std_{ext} and std_{true}. Now, let the means of m_{int}, m_{ext} and m_{true} be denoted M_{int}, M_{ext} and M_{true}. The means shown in Table 1 are then M_{int}, M_{ext} and M_{true}, while the standard deviations are the standard deviations of M_{int}, M_{ext} and M_{true}. These are calculated according to the formula $STD = std/\sqrt{1000}$.

M_{int}:	0.1411
M_{ext}:	0.1802
M_{true}:	0.1853
STD_{int}:	0.0015
STD_{ext}:	0.0020
STD_{true}:	0.0006

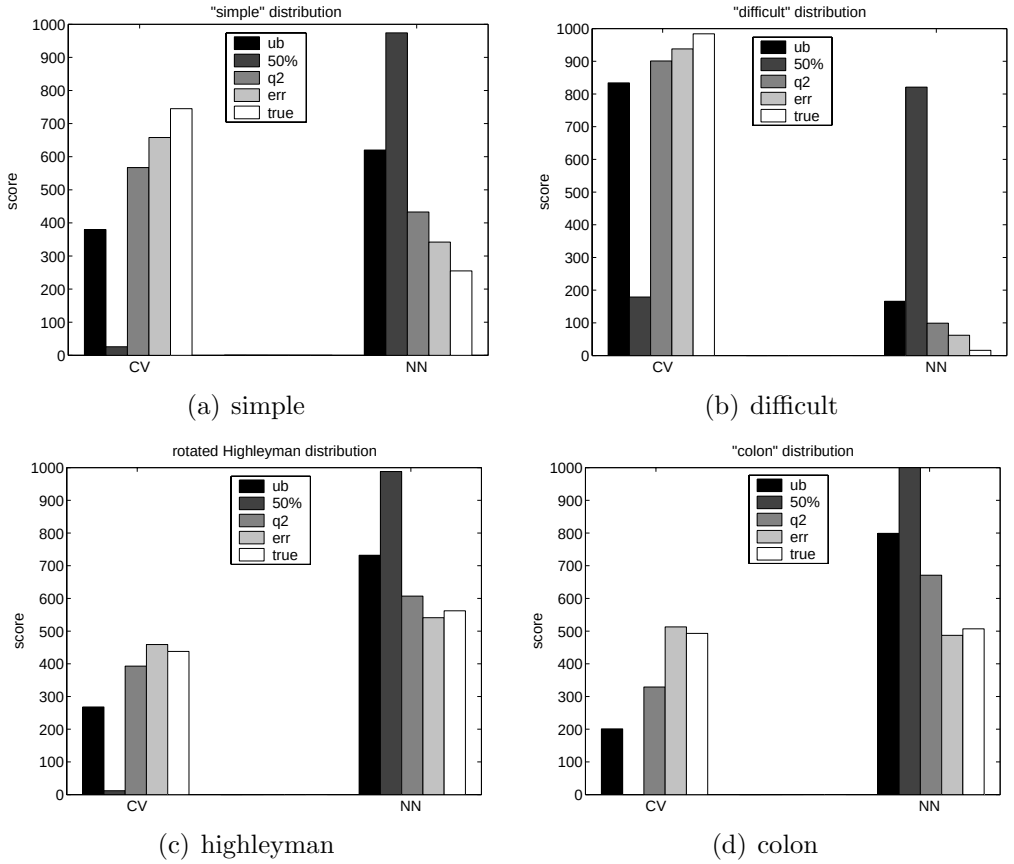
Tab. 1: Mean values and standard deviations of average error rates.

It should be noted that the standard deviations presented in Table 1 are standard deviations of the mean and not the standard deviations of the distributions in Fig. 4. As expected and as can be seen from Fig. 4(a) and 4(c), as well as in Table 1, the internal average error estimate M_{Int} have a large optimistic bias while the external average error estimate M_{Ext} is almost identical to the true average error estimate M_True. The results in Fig. 4(b) show that the internal and external estimates have a strong positive correlation, but the large variances imply that the result from a single study might be quite misleading, even if it is unbiased. For example, as can be seen in Fig. 4(a), while the vast majority of the average of 'true' error rate estimates are observed on the interval [0.16,0.22] the corresponding 'external' error rate estimates are observed on the interval [0.05,0.30].

4.2 Comparison Test

As described in Section 3.7, simulations were carried out in order to compare an algorithm for selecting between four different classifiers with a simple 3-NN classifier. The two procedures were compared with respect to several different error measures, as described in Material and Methods. Comparisons

were done for 1000 data sets from each of four different distribution types exemplified in Fig. 1.



(a) simple (b) difficult

(c) highleyman (d) colon

Fig. 5: Comparison test applied to the different distributions.

The results are shown in Fig. 5. The score values in the figures refer to how many times each of the two algorithms performed best according to the different performance measures.

When applied to the "simple" distribution (see Fig. 5(a) and 1(a)), (ub) and (50%) gave the nearest neighbour algorithm (a2) the highest score, while (q2) and (err) gave the CV optimization algorithm (a1) the highest score. The score from (true) showed that CV algorithm performed the best on this problem.

For the data set from the "difficult" distribution (see Fig. 5(b) and 1(b)), all performance measures except (50%) gave (a1) the highest score. For

the (50%) measure, the score for (a2) was much higher than that for (a1). Despite this, (true) clearly shows that (a1) performed the best.

On the rotated Highleyman data set (see Fig. 5(c) and 1(c)), (a2) got the highest score. The hold-out test on the large test set, (true), confirms that (a2) actually performed the best on this underlying distribution.

Similarly, on the "colon" data set (see Fig. 5(d) and 1(d)), most performance measures gave NN the highest score. An exception here was the small hold-out test, which by a small margin gave the CV algorithm, (a1), the highest score. The score from (true), however, was highest for (a2).

5 Discussion

5.1 CV simulations

In this work a classifier is seen as a black box - the word "classifier" is to be understood as the sentence "something that outputs class labels when presented with unlabeled data". Therefore, an algorithm which chooses between different classifiers based on the training data is also a classifier. (Even an algorithm that randomly assigns class labels to new data is by this definition viewed as a classifier!)

For the double CV-loop, several sets of error estimates are produced. The inner loop renders CV-estimates of the error rates of all the different classifiers (all choices of k). The error estimate that the inner loop outputs in each iteration, however, is only the lowest one. When forming a distribution from these error rate estimates, this distribution will describe the error rate of the classifier constituted by the entire inner loop - not that of any of the particular classifiers contained in it.

The results from the double CV simulations show that a bias is in fact induced when using cross CV to choose between several alternative classifiers, especially for small data sets. When neglecting to use independent testing, a positive bias of 0.0442 units or near 24% of M_{true} is induced in M_{int} (see Fig. 4(c) and Table 1). Recalling that a CV error rate estimate is really no more than a sample from an error rate distribution with a large unknown variance, the cumulative effects of the positive bias and distribution variance make an estimate produced this way practically useless.

Fig. 4(a) shows that a remedy for the bias-part of the problem is to test the resulting classifier on previously unused data in a double CV-loop procedure. The deviation from the "true" error rate is much smaller here, and the 95% confidence intervals for M_{ext} and M_{true} overlap (see Table 1). The error estimates produced in the external loop are thus approximately unbiased.

The need for independent test data, however, implies that data is consumed by the selection process. If we want to increase our chances of performing well, we must use some data to perform some kind of optimization process. This means that if we also want to have a reasonably certain estimate of how well our classifier is going to perform, the true performance will decrease compared to a situation in which no performance estimate is required.

Finally, even if we use some data for selection, it is important to note that the double CV-loop does not resolve the variance problem. In other words, although we have an unbiased error estimate, if the independent test set is small, the uncertainty about the true performance is surprisingly large.

5.2 Comparison Test

The results from the comparison algorithm show that for some target functions, choosing a relatively robust classifier is in fact preferable over trying to optimize the selection. Examples of this can be seen in Fig. 5(c) and 5(d).

It is also shown that for some problems and performance measures, when optimizing choice of classifier or parameters based on CV, the decrease in certainty about the performance induced by loss of data will obscure the fact that it really performs quite well compared to an algorithm lacking optimization ability. This is examplified in Fig. 5(a) and 5(b).

It is also interesting to note that the (50%) measure (see Fig. 5) without exception gives (a2) the highest score. The (50%) seems intuitively reasonable - a minimum requirement to place on a classifier is that it performes better than random guessing. However, in the cases considered here, the probability mass above 50% in the calculated pdf is in almost every case far below 1/1000 of the total mass.

One must remember that the different Bayesian posteriors resulting from the comparison test describe our uncertainty about a classifier's performance. This gives information about what is reasonable for us to assume based on

available knowledge. It says relatively little about any actual true performance of a classifier. The reason for this is of course that a simple holdout test also tells us relative little about the true performance.

5.3 Why Does the Bias Occur?

Suppose we have a procedure P that chooses between three different classifiers based on the CV-error estimate. If we repeat the process, three CV error estimates will be produced for each iteration. With a frequentistic approach, we may say that these error rate estimates, after an infinite number of iterations, will form three error rate estimate distributions. We denote these $\tilde{p}(\hat{e}|n)$, where n is the classifier index.

Now suppose that for each iteration, P outputs the lowest error rate estimate, along with its corresponding classifier. Again, if we repeat P an infinite number of times, and collect the error rate estimates that P outputs, we are able to form three new distributions, one for each choice of classifier. We denote these $\tilde{\tilde{p}}(\hat{e}|n)$.

If we substituted the CV procedure for a holdout test on an infinite number of test examples, we would end up with three distributions of the true error rate, one for each classifier. These will be denoted $p(e|n)$.

Depending on the underlying distribution from which the data is gathered, P will have different probabilities of choosing each of the three classifiers. Continuing our hypothetical experiment, we can easily determine these probabilities by counting how many times each classifier is chosen. This results in three probabilities, denoted α_n.

If we are interested in gaining knowledge about the performance of P, the ideal way would be to calculate the true error rate distribution, $p(e)$, according to:

$$p(e) = \sum_{n=1}^{3} \alpha_n \cdot p_n(e|n) \tag{5}$$

Let us now suppose we lack an infinite data set for holdout testing. The intuitive alternative is then to form a distribution from the error rate estimates that P outputs. This is equivalent to:

$$p_a(\hat{e}) = \sum_{n=1}^{3} \alpha_n \cdot \tilde{\tilde{p}}_n(\hat{e}|n) \tag{6}$$

However, since for each choice of n, $\tilde{\tilde{p}}(\hat{e}|n)$ are built on only the lowest error rate estimates from each classifier, they must be biased. This is easily realized when comparing them to $\tilde{p}(\hat{e}|n)$. For each n, $\tilde{p}(\hat{e}|n)$ is just a pdf of CV estimates for a single classifier, a type of estimate which is commonly known to be unbiased. This implies that $\tilde{\tilde{p}}(\hat{e}|n)$ is biased. It also implies that an unbiased error rate estimate distribution for P could be obtained according to:

$$p_b(\hat{e}) = \sum_{n=1}^{3} \alpha_n \cdot \tilde{p}_n(\hat{e}|n) \tag{7}$$

Naturally, this thought experiment is of little practical use, since in real-life situations only one data set is usually available. It is, however, of theoretical importance for understanding the reason for bias in the type of situations considered in this work.

5.4 Generality of the Results

One implication of the NFL is that there is no such thing as a good classifier [6], [7]. A classifier is only good in relation to a particular pair of data sets, one on which the algorithm is trained, and one to which it is applied. Thus, a more convenient way of thinking of this issue is to do it in terms of classifier-problem pairs, with particular error rates associated with them. So, from an error rate point of view, it is not important if we modify the classifier or the problem - both kinds of alteration will result in a migration from one classifier-problem pair to another.

The explanation of why bias in optimization by CV occurs implies that the bias is a direct consequence of how the different distributions interrelate. Given that the bias can be described completely by the relations between the different distributions, it seems reasonable to assume that as long as our different choices of parameters, be it parameters for the classifier for feature selection or for feature extraction, produce different results, it is really

unimportant which parameters to tune. The bias will occur regardless of how the distributions were produced.

This shows that the results in this work holds equally well for procedures choosing between different parameters for feature selection or extraction, between different parameters for a certain classifier, or between several different classifiers.

6 Conclusions

Bias in error rate estimation occurs when using CV for optimizing classifier, parameter or feature selection, if improper test procedures are used. Obtaining unbiased error rate estimates consumes data, a fact which in turn adds to the uncertainty already inherent in the CV procedure. A way of mathematically treating this uncertainty is by means of Bayesian inference. This approach reveals that in some situations, a pre-defined choice of classifier is in fact preferable to optimization, while in others, the uncertainty induced by loss of data can be misleading in evaluating an optimization procedure.

7 Acknowledgements

Mats Gustafsson and Anders Isaksson for great supervision and encouragement, Karin Holmström for productive discussions and ideas, professor Jan Komorowski for kindly providing workspace and cpu-time, and all personel att LCB for a stimulating and friendly environment.

References

[1] Webb, A., *Statistical pattern recognition (2nd ed).* John Wiley & Sons, New York 2002.

[2] Bishop, C. M., *Neural networks for pattern recognition.* Oxford University Press, Oxford 2000.

[3] Simon, R., Radmacher, M. D., Dobbin, K., McShane, L. M., Pitfalls in the use of microarray data for diagnosic and prognostic classification. *J Nat Cancer Inst* 2003;**95**:14-18.

[4] Bahsin, M., Raghava, G. P., SVM based method for predicting HLA-DRB*0401 binding peptides in antigen sequence. *Bioinformatics* 2004;**20**:421-423.

[5] Ambroise, C., McLachlan, G. J., Selection bias in gene extraction on the basis of microarray gene-expression data. *PNAS* 2002;**99**:6562-6566.

[6] Wolpert, D. H., *The supervised learning no-free-lunch theorems.* http://ic.arc.nasa.gov/ic/projects/bayes-group/people/dhw/ 2001 (10 Dec. 2004)

[7] Duda, R. O., Hart, P. E., Stork, D. G., Feature selection for high-dimensional genomic microarray data *Pattern Classification (2nd ed).* John Wiley & Sons, New York 2001

[8] Duin, R. P. W., de Ridder, D., Juszczak, P., Paclik, P., Pekalska, E., Tax, D. M. J. PRTools toolbox. Delft University of Technology 2004. www.prtools.org

[9] Lissack, T., Fu, K. S., Error estimation in pattern recognition via L-distance between posterior density functions, *IEEE Trans Inform Theory* 1976;**22**:34-45

[10] Mullin, M., Sukthankar, R., Complete cross-validation for nearest neighbour classifiers. *Proc ICML* 2000;639-646.

[11] Anders Isaksson, personal correspondance.

Read Litterature

van't Veer, L. J., Dai, H., van de Vijver, M. J., He, Y. D., Hart, A. A. M., Mao, M., Peterse, H. L., Gene expression profiling predicts clinical outcome of breast cancer. *Nature* 2002;**415**:530-536

Liu, J., Rost, B., Sequence-based prediction of protein domains. *Nucleic Acids Res* 2004;**32**:3522-3530.

Xing, E. P., Jordan, M. I., Karp, R. M., Feature selection for high-dimensional microarray data *Proc ICML* 2001;601-608.

Shannon, C. E., A Mathematical Theory of Communication. *Bell System Technical Journal*, 1948;**27**:379-423 & 623-656.