Density Preserving Sampling: Robust and Efficient Alternative to Cross-validation for Error Estimation

Marcin Budka, *Member, IEEE,* and Bogdan Gabrys, *Senior Member, IEEE*

***Abstract*—Estimation of the generalization ability of a classification or regression model is an important issue, as it indicates expected performance on previously unseen data and is also used for model selection. Currently used generalization error estimation procedures like cross–validation (CV) or bootstrap are stochastic and thus require multiple repetitions in order to produce reliable results, which can be computationally expensive if not prohibitive. The correntropy–inspired Density Preserving Sampling procedure (DPS) proposed in this paper eliminates the need for repeating the error estimation procedure by dividing the available data into subsets, which are guaranteed to be representative of the input dataset. This allows to produce low variance error estimates with accuracy comparable to 10 times repeated cross–validation at a fraction of computations required by CV. The method can also be used for model ranking and selection. This paper derives the Density Preserving Sampling procedure and investigates its usability and performance using a set of public benchmark datasets and standard classifiers.**

***Index Terms*—Error estimation, model selection, sampling, cross–validation, bootstrap, correntropy.**

I. Introduction

ESTIMATION of the generalization ability of a classification or regression model is an important issue in machine learning, especially that it is independent of the actual model used. Generalization accuracy estimates act not only as the indicators of expected performance on previously unseen data, but are also routinely used for model ranking and selection [1].

In contrast to the large number of various regression and classification methods currently in use, there is only a handful of model independent generalization error estimation techniques. The most popular of them are cross–validation [2] dating back to 1968, and bootstrap [3] developed in 1979. These techniques, and especially cross–validation are being used even more willingly and blindly after the publication of a seminal paper by Kohavi in 1995, presenting a comparative study of bootstrap and cross–validation [4], and currently estimated to have almost 2800 citations [5].

The basic idea, shared by all generalization error estimation methods, is to reserve a subset of available data to test the model after it has been trained using the remainder of the dataset[1]. The main difference between various techniques is the way the generalization error is calculated, the size of the subset reserved for testing or whether the procedure is repeated multiple times or not. They have however also something in common, and that is the way in which the testing subset is generated – random sampling. Although the stochastic nature of bootstrap and cross–validation ensures that in the limit they would both converge to a true value, this may also lead to large variations in the estimate between consecutive runs, making the results unreliable. This effect can be alleviated to a large extent by repeating both procedures multiple times, which however significantly increases the computational demands.

A good test set should be independent of the training data and representative of the population from which it has been drawn. While random sampling meets the first requirement, it does not guarantee the representativeness. Stratified sampling approaches [4] address this issue by trying to increase the representativeness at the expense of independence, and are able to achieve better results than their non–stratified counterparts.

Building upon the success of stratified sampling approaches, we propose a Density Preserving Sampling procedure (DPS), which goes a step further and samples the data at the level of individual clusters to enforce representativeness of the test set. This is achieved by optimizing a dataset similarity index inspired by the correntropy, a non–parametric similarity measure of two random variables [7]. According to the experiments, DPS produces accurate generalization estimates, requiring a fraction of computations when compared to cross–validation.

This paper extends [8] among others by providing better theoretical justification of the method, considerably extending its experimental evaluation (additional datasets, comparison with bootstrap), investigating the problem of DPS-guided model selection and building of ensemble models. The paper is organized as follows. In Section II the problem of estimation of generalization error is introduced, together with standard techniques and criteria of their evaluation. Section III describes the concept of Information Theoretic Learning and correntropy, used in the novel Density Preserving Sampling procedure derived in Section IV. The experimental results, including evaluation of DPS in terms of bias and variance as well as its usability for model selection are given in Section V. Finally, the discussion and conclusions can be found in Sections VI and VII respectively.

II. Generalization error estimation

Generalization error is the error of a predictive model on previously unseen data, generated from the same distribution as the data used to develop it [1]. Low generalization error is thus a sign of a good match between the model and the problem, and lack of overfitting.

It is impossible to obtain a closed form solution for calculation of the generalization error or even for calculation of tight

M. Budka and B. Gabrys are with the Smart Technology Research Centre, Bournemouth University, School of Design, Engineering and Computing, Poole House, Talbot Campus, Fern Barrow, Poole BH12 5BB, UK.
Email: mbudka@bournemouth.ac.uk / bgabrys@bournemouth.ac.uk

[1] There also exist methods for in-sample error estimation, which however are not of general purpose (e.g. [6] for Support Vector Machines).

bounds for the error, in all but the simplest cases [9]. The only practical solution is to estimate the generalization error from all available i.i.d. (independently and identically distributed) data samples by splitting them into training and validation sets [10]. For the error estimate to be meaningful both these datasets should be representative of the true distribution, so the way in which the data is split plays a crucial role.

A. Hold–out and random subsampling

The simplest and the least computationally expensive way to estimate the generalization error is the hold–out method [11], in which the data is split randomly into two parts: the training set and the hold–out set, in a priori chosen proportions. The model is then trained using the training dataset and its error on the hold–out data becomes an estimate of the generalization error. The drawback of the hold–out method is that unless both datasets are large enough (which is a vague term in itself), different estimates will be obtained from one run to another. Hence in random subsampling [10], the hold–out procedure is repeated multiple times and the results are averaged. This however still does not guarantee that all instances will at some point be used for training nor that none of the classes will be over/under–represented in the hold–out set [4]. To circumvent these issues, more advanced resampling techniques have been developed. Yet, the hold–out method is still used when dealing with large datasets if other techniques become untractable. It is also sometimes assumed that more advanced resampling techniques are simply not needed for large amounts of data.

B. Cross–validation

Cross–validation (CV) is a widely used standard statistical technique for estimation of model generalization ability, applied with a great success to both classification and regression problems [1]. In k–fold cross–validation the whole available dataset is randomly divided into k approximately equal subsets. Each of these subsets or folds is then in turn put aside as validation data, a model is built using the remaining $k-1$ folds and tested on the validation fold. The error estimate is then calculated as a mean of all validation errors, while their standard deviation can be used to approximate the confidence intervals of the obtained estimate. The whole procedure thus requires development of exactly k models. Since the results obtained in this setting are also likely to vary from one run to another, the procedure is repeated multiple times for various random splits, and the results are averaged.

The most often used variants of cross–validation are:

- Leave–one–out cross–validation in which a single instance is used as a validation set. This produces unbiased but high–variance error estimates and can be computationally prohibitive for large datasets.
- Repeated 10–fold cross–validation, which often is a good compromise between speed and accuracy.
- Repeated 2–fold cross–validation, which is an approximation of the bootstrap method [11].

In order to improve the accuracy of the estimates obtained, a stratified CV approach is used in practice, which samples the data in a way that approximates the percentage of each class in every fold [4]. For regression problems, stratified CV produces folds with equal means of the target variable [10].

C. Bootstrap

Bootstrap is a second commonly used generalization error estimation procedure [1], especially useful when dealing with small datasets [11]. Given an input dataset of size m, the method performs uniform sampling with replacement to produce a training set of the same size. The instances not selected during the sampling procedure become the test set. The probability of each instance ending up in the test set is $\left(1-\frac{1}{m}\right)^m \approx e^{-1} \approx 0.368$, while the probability of ending up in the training set is $1-0.368=0.632$. Hence the method is also often called the '0.632 bootstrap' [11]. Since the error estimate obtained using test data only would be overly pessimistic (only 63.2% of instances are used for training), to compensate for this effect it is calculated as $error = 0.632 \times e_0 + 0.368 \times e_{bs}$, where e_0 is the error rate obtained from bootstrap sets not containing the point being predicted (test set error) and e_{bs} is the error obtained on the bootstrap sets themselves, both averaged over all instances and bootstrap samples. The more times the whole process is repeated, the more accurate the estimate. A detailed treatment of the bootstrap methods can be found in [12].

Techniques like bootstrap and CV have been developed primarily to address the situations when data is scarce and one cannot afford a separate hold-out set. In the case when data is abundant and its distribution does not undergo changes over time, a single stratified random split is usually able to provide the required level of representativeness.

A comparative study of CV and bootstrap for a set of standard benchmark datasets can be found in [4]. A number of follow-up studies for microarray data [13], synthetic data [14] and regression problems [15] also exist.

D. Bias and variance of error estimation methods

The bias of an error estimation method is the difference between the expected value of the error and the estimated value [4]. For an unbiased estimator, this difference is equal to zero. Bias can also be either positive or negative. In the former case, the estimate is said to be overly optimistic, as the estimated error is lower than the expected error. Negative bias on the other hand leads to overly pessimistic error estimates.

Low bias on its own does not guarantee good performance of the model. There is another important parameter – the variance, which measures the variability of the error estimate from one run to another. In the case of subsampling methods discussed in this paper, the variability is usually approximated by the expected standard deviation of a single accuracy estimation run [4]. A good generalization error estimator should thus have low bias and low variance. Unfortunately in practice it is usually difficult to achieve both at the same time, leading to so called bias–variance trade–off [1].

III. Representative sampling

Both cross–validation and bootstrap, described in Sections II-B and II-C are stochastic methods. The immediate consequence is that the results can vary a lot from one repetition to another and there is no guarantee that the datasets obtained by splitting the original data are representative, which is a necessary condition for obtaining accurate error estimates. Thus in order to obtain reliable results, averaging over multiple iterations is required. In general, the more times the procedure is repeated the better, as in the limit both methods would converge to the true error values. For $k-$fold CV using $m-$element dataset this could mean averaging over all $\binom{m}{m/k}$ ways of choosing m/k instances out of m (complete cross–validation [4]), which quickly becomes untractable. There is however another, often overlooked possibility – intelligent sampling aiming at producing only representative splits.

From statistics, a random sample is considered representative if its characteristics reflect those of the population from which it is drawn [16]. Since these characteristics are naturally reflected by the probability density function (PDF), the more similar the distribution of the sample to the distribution of the population, the more representative this sample is. Hence a sampling procedure aiming at maximization of some similarity measure between PDFs of the two samples should intuitively ensure their mutual representativeness.

There are many PDF divergence measures in the literature [17]. Perhaps the best known of them is the Kullback–Leibler divergence [18] and its two modifications i.e. Jeffrey's and Jensen–Shannon divergences [19]. Although most of these measures have strong theoretical foundations, there are usually no closed–form solutions to calculate them, they are difficult to estimate from samples smaller than thousands of instances, and their direct optimization is even more challenging [20], [21]. However, a recently developed Information Theoretic Learning framework [22] can provide another solution.

A. Information Theoretic Learning (ITL)

Information Theoretic Learning is a procedure of parameter adapting using an information theoretic criterion [22]. Application of the information theory to learning problems is however not straightforward. The main issue is the omnipresent 'learning from exemplars' paradigm, while the information theory in its traditional form is only able to deal with PDFs given in analytic forms [23]. The cornerstone of ITL is thus an alternative, easily calculable definition of entropy – the Renyi's quadratic entropy: $H_{R_2} = -\log \int p(y)^2 \, dy$. An important property of Renyi's entropy is that its extrema overlap with the extrema of Shannon's entropy [22], so both definitions are equivalent for the purpose of optimization.

Denoting by $G(y,\ \sigma^2 I)$ a spherical Gaussian kernel centered at y with a diagonal covariance matrix $\sigma^2 I$, the PDF can be estimated using the Parzen window method [1] as $p(y) = \frac{1}{N}\sum_{i=1}^{N} G(y - y_i,\ \sigma^2 I)$. Using the convolution property of the Gaussian kernels, the Renyi's entropy becomes:

$$H_{R_2}(y) = -\log \int p(y)^2 \, dy = -\log V(y) \tag{1}$$

$$V(y) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(y_i - y_j,\ 2\sigma^2 I) \tag{2}$$

where $V(y)$ is an averaged sum of all pairs of interactions between all pairs of instances called the Information Potential.

B. Auto– and cross–correntropy

A Generalized Correlation Function (GCF) for a stochastic process x_t has been defined in [24] as:

$$V_X(t_1, t_2) = E[< \phi(x_{t_1}), \phi(x_{t_2}) >] = E[k(x_{t_1}, x_{t_2})] \tag{3}$$

where E stands for the expected value, ϕ denotes some kernel induced transformation and k is a kernel function, assumed to be Gaussian from now on. The GCF estimator not only conveys information about autocorrelation but also about the structure of the dataset, as its mean value for non–zero lags converges asymptotically to the estimate of the Information Potential calculated using Renyi's quadratic entropy [25]. For this reason the function has been named auto–correntropy.

The idea of auto–correntropy has been further developed in [7] for a general case of two arbitrary random variables. The new measure, named cross–correntropy (or correntropy) is defined for variables X and Y as:

$$V_{XY}(X, Y) = E[< \phi(X), \phi(Y) >] = E[k(X, Y)] \tag{4}$$

The correntropy can be used as a measure of similarity between X and Y but only in the neighbourhood of the joint space. This results from the restriction of Gaussian kernels, which have high values only along the $x \approx y$ line with exponential fall off otherwise. The size of this neighbourhood is therefore controlled by the kernel width parameter σ. Correntropy can thus also be defined as the integral of the joint PDF along the $x = y$ line, i.e. $V_{XY}(X, Y) \approx \int p(x, y) \mid_{x=y=u} du$. By plugging the Parzen window estimate of the joint PDF $p(x, y) \approx \frac{1}{N}\sum_{i=1}^{N} G(x - x_i, \sigma^2 I) G(y - y_i, \sigma^2 I)$, integrating along the $x = y$ line and using the convolution property of Gaussians again, the estimate of correntropy finally becomes:

$$V_{XY}(X, Y) \approx \frac{1}{N} \sum_{i=1}^{N} G(x_i - y_i, 2\sigma^2 I) \tag{5}$$

Correntropy can be regarded as the PDF of equality of two variables in the neighbourhood of the joint space, of the size determined by the kernel width parameter σ [7], [26]. The measure has many interesting properties and one of them is that for independent X and Y it can be approximated by the Information Potential formula similar to (1) and named Cross Information Potential [26]. Correntropy has been successfully used as a localized, outlier–resistant similarity measure for many supervised learning applications [27], [26].

IV. Density Preserving Sampling (DPS)

Since the correntropy can be used to measure similarity between two random variables, it can also be used to measure the 'representativeness' of a sample. It should thus be possible to use correntropy as an optimization criterion, guiding the sampling process in order to split a given dataset into two or more maximally representative subsets. Moreover due to the properties of (5) discussed in more detail in Section IV-B, it is possible to devise an optimization procedure independent of the Gaussian kernel width σ. This fact, together with the conclusions drawn from a comprehensive evaluation of various divergence measures in the context of representative sampling presented in [20], is the main reason for considering correntropy–based objective function.

A. Correntropy-inspired Similarity Index (CiSI)

Equation (5) defines the estimator of correntropy between two random variables or datasets X and Y as the value of a Gaussian kernel centered at $(x_i - y_i)$ averaged over all N instance pairs. There are thus three requirements for calculation of the correntropy to be possible: the datasets must (1) be ordered, (2) have the same dimensionality and (3) have the same cardinality. Put another way, for equation (5) to be a meaningful estimator of correntropy, a random process generating samples according to the joint PDF of X and Y needs to be defined, which in our case is not straightforward.

While the second requirement is irrelevant for sampling, as each subset of objects necessarily needs to have the same dimensionality as the set from which it has been selected, the remaining two requirements remain valid. For some applications like e.g. supervised learning, all the above requirements are met automatically – if A denotes the output of a mapper and B denotes the target value, $|A| = |B|$ and a_i is the prediction of b_i. In sampling however in general one cannot expect the instances to be ordered, which means that it is not obvious on the difference of which instances to center the Gaussians (the joint PDF is not defined). The datasets may also have different cardinalities e.g. when one wants to calculate the objective between the original dataset and its subset.

To address the ordering issue, we make sure that every Gaussian in (5) is centred at $(x_i - y_j)$, which maximize its value. This is achieved by selecting both x_i and y_j so that they are as close to each other as possible. The generalized, instance ordering insensitive version of (5), now called the Correntropy-inspired Similarity Index (CiSI) is thus given by:

$$CiSI(X,Y) \approx \frac{1}{N} \sum_{i \in (1..N)} G(x_i - y_j, 2\sigma^2 I)$$
$$i,j = \operatorname*{argmin}_{i,j} \parallel x_i - y_j \parallel, \quad j \in J_{avail} \qquad (6)$$

where $\parallel \cdot \parallel$ denotes the Euclidean norm and the set J_{avail} contains indices of y which haven't yet been used, to ensure that each y_k is used only once.

When the datasets have different cardinalities, i.e. without loss of generality if $N_X > N_Y$, the above approach will terminate after N_Y instances are processed. To avoid this, a new dataset Y_N is created by duplicating the original Y dataset $\lceil N_X/N_Y \rceil$ times. CiSI is then calculated between X and Y_N and the calculation will terminate after exactly N_X steps.

The values of CiSI reported in this paper have been normalized to the $[0, 1]$ range. However, these values are not comparable between different datasets as their absolute difference can be made almost arbitrarily large by manipulating the Parzen window width σ. For this reason the CiSI values given should rather be perceived as ranks on an ordinal scale.

B. CiSI based sampling procedure

In this section we propose a CiSI–based, hierarchical, density preserving splitting procedure. Since the generalized function of (6) is not differentiable with respect to i and j (the only variables that can be manipulated within the splitting process), gradient driven optimization is not straightforward. We have thus reverted to a greedy, locally optimal approach, which in many cases prove to work surprisingly well [28].

As CiSI is being estimated by a normalized sum of Gaussians, it reaches a maximum when all its components reach their maximal values. Since a Gaussian function peaks at 0 regardless of the value of σ, and it is piecewise monotonic and symmetric[2], the closer x_i and y_j are in (6), the higher $CiSI(X,Y)$ will be. This suggests a σ–independent, iterative, binary splitting procedure of dataset Z into subsets X and Y, which at each step selects instances z_i and z_j so that:

$$i,j = \operatorname*{argmin}_{i,j} \parallel z_i - z_j \parallel \qquad (7)$$

and then adds them to the sets X and Y, so that $X = X \cup z_i$ and $Y = Y \cup z_j$ or the other way round, removing them from dataset Z at the same time. The above procedure aims at directly maximizing $CiSI(X,Y)$, that is the similarity index between the two new datasets. Due to the way CiSI is calculated for sets with various cardinalities however, it also indirectly maximizes $CiSI(X,Z)$ and $CiSI(Y,Z)$. As a result, newly obtained datasets are splits with PDFs maximally similar to each other and to the PDF of the original dataset. To obtain more splits, the procedure can be repeated by splitting datasets X and Y again, which will produce 4 splits and so on. The total number of splits is thus always a power of 2.

The instances z_i and z_j can be added to the sets X and Y arbitrarily or not. In our approach we have devised a procedure in which the two objects are distributed in a way that maximizes the average coverage of the input space by both splits. Denoting by d_{kV} the average Euclidean distance between instance z_k and all instances in set V, the rules are:

$$d_{iX} + d_{jY} \geq d_{jX} + d_{iY} \Rightarrow X = X \cup z_i,\ Y = Y \cup z_j \qquad (8)$$
$$d_{iX} + d_{jY} < d_{jX} + d_{iY} \Rightarrow X = X \cup z_j,\ Y = Y \cup z_i \qquad (9)$$

For classification problems, the splitting procedure can be executed in a supervised or unsupervised mode. In the former case, the algorithm takes advantage of the class labels supplied with the data by considering each class in separation from the rest. We refer to this approach as DPS–S. In the unsupervised mode, the class labels are ignored, so the procedure is purely density–driven and has been called DPS–U.

In current implementation, if the classes are too small to be divided into a given number of subsets, DPS–S falls–back to DPS–U. Since the splitting procedure is hierarchical, most datasets can be divided using the supervised approach up to some point, before the unsupervised procedure takes over.

When using a k-d tree [29] to perform the nearest neighbour search the computational complexity of DPS-U is of the order $O(N \log N)$ on average and $O(N^2)$ in the worst-case scenario. The memory complexity is of the order $O(N)$. For DPS-S, these become $O(\sum_i N_i \log N_i)$, $O(\sum_i N_i^2)$ and $O(\sum_i N_i)$ respectively, where N_i is the cardinality of the i^{th} class. Note, that even in the worst-case scenario these are still negligible when compared to the complexity of most learning algorithms.

[2] In fact, any kernel function which has these properties could be used instead, leading to exactly the same results (e.g. Triangular, Epanechnikov).

V. Experiments

The experiments have been conducted on 27 publicly available datasets using a total of 16 different classifiers. The datasets used come from [30], [31] and [32] and their details are given in Table I. The star symbol in the '#obj/attr' column denotes the number of instances actually used in the experiment, sampled randomly from the whole, much bigger dataset in order to keep the experiments computationally tractable. The classifiers used are implemented within the PRTools toolbox and their list is given in Table II. It is worth to mention that starting from version 4.1.10, the DPS procedure proposed in this paper has been included into the PRTools toolbox as an alternative to the stratified CV.

The experiments were designed to (a) compare the error estimation accuracy of stratified CV, bootstrap and DPS, (b) test the stability of both error estimators, (c) test applicability of DPS to the classifier selection process, (d) investigate the possibility to reliably estimate the generalization error using a single DPS fold only, thus reducing the computational requirements by another order of magnitude, and (e) examine the behavior of DPS in the context of ensemble models.

We have followed a similar approach to that outlined in [4]. For each dataset a stratified random subsampling procedure has been repeated 100 times, resulting in 100 random divisions of the dataset into a training part $(2/3)$ and independent test data $(1/3)$. The training part was then used to estimate the generalization error using CV and DPS for each classifier, while the independent test part has been used to calculate the 'true' generalization error, once again for each classifier in turn. The true generalization error then served to calculate the bias of each estimate, while the generalization error estimates of a single estimation run have been used to calculate the variance. Finally, the results have been averaged over all 100 runs of the random subsampling procedure.

The CV estimate was calculated within a 10 times repeated 8–fold scheme. We provide the average results for all 10 iterations as well as the result of the best and worst single run in terms of bias/variance to emphasize how unstable the CV error estimates can be. Three 8–fold DPS estimates are also given – DPS–S (using class labels), DPS–U (ignoring class labels) and DPS–SU (averaged over the errors of classifiers trained on DPS–S and DPS–U folds). For the comparison with DPS to be fair from the computational point of view, the bootstrap estimate was also calculated on 8 bootstrap samples.

A. Toy problems

The analysis starts with two synthetic datasets first used in [33] and [34]. The datasets were chosen because they are two–dimensional, allowing for easy visualisation, and were used in our previous studies due to their well known properties.

1) Cone–torus dataset: Cone–torus is a synthetic 2 dimensional dataset consisting of 3 classes, with data points generated from 3 differently shaped distributions: a cone, half a torus, and a Gaussian. A scatter plot of the dataset is given in Figure 1(a). Figure 2 depicts scatter plots of 8 DPS–S folds, while in Figure 3, 8 CV folds generated during a single random run are given. Note, that in case of DPS, the classes tend to preserve their shapes – the half torus for example is clearly visible in 7 out of 8 folds, while for CV only in 4 or 5. This is also well reflected by the mean value of CiSI between all 8 folds and the original dataset, which is 0.81 for DPS and 0.71 for CV averaged over 10 runs ($\sigma = 0.12$).

The decision boundaries for the *qdc* classifier trained on each of 8 folds in turn, superimposed on the original dataset have been given in Figure 4. The black solid line represents the boundaries of a classifier trained using the DPS–S folds, while

TABLE I
Dataset details

abbr	name	source	#obj/attr	#class
azi	Azizah dataset	PRTools	291/8	20
bio	Biomedical diagnosis	PRTools	194/5	2
can	Breast cancer Wisconsin	UCI	569/30	2
cba	Chromosome bands	PRTools	1000*/30	24
chr	Chromosome	PRTools	1143/8	24
clo	Clouds	ELENA	1000*/2	2
cnc	Concentric	ELENA	1000*/2	2
cnt	Cone–torus	[33]	800/3	2
dia	Pima Indians diabetes	UCI	768/8	2
ga2	Gaussians 2d	ELENA	1000*/2	2
ga4	Gaussians 4d	ELENA	1000*/4	2
ga8	Gaussians 8d	ELENA	1000*/8	2
gla	Glass identification data	UCI	214/10	6
ion	Ionosphere radar data	UCI	351/34	2
iri	Iris dataset	UCI	150/4	3
let	Letter images	UCI	1000*/16	26
liv	Liver disorder	UCI	345/6	2
pho	Phoneme speech	ELENA	1000*/5	2
sat	Satellite images	UCI	1000*/36	6
seg	Image segmentation	UCI	1000*/19	7
shu	Shuttle	UCI	1000*/9	7
son	Sonar signal database	UCI	208/60	2
syn	Synth–mat	[34]	1250/2	2
tex	Texture	ELENA	1000*/40	11
thy	Thyroid gland data	UCI	215/5	3
veh	Vehicle silhouettes	UCI	846/18	4
win	Wine recognition data	UCI	178/13	3

TABLE II
Classifier list

name	description
fisherc	Fisher's Linear Classifier
ldc	Linear Bayes Normal Classifier
loglc	Logistic Linear Classifier
nmc	Nearest Mean Classifier
nmsc	Nearest Mean Scaled Classifier
quadrc	Quadratic Discriminant Classifier
qdc	Quadratic Bayes Normal Classifier
udc	Uncorrelated Quadratic Bayes Normal Classifier
klldc	Linear Classifier using KL expansion
pcldc	Linear Classifier using PC expansion
knnc	K–Nearest Neighbor Classifier
parzenc	Parzen Density Based Classifier
treec	Decision Tree Classifier
naivebc	Naive Bayes Classifier
nusvc	Support Vector Classifier with linear kernels
rbnc	Radial Basis Function Neural Network Classifier

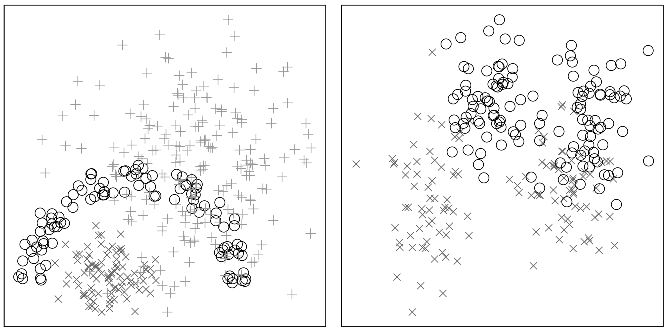

(a) Cone–torus (b) Synth–mat

Fig. 1. Scatter plots of the synthetic datasets used in the experiments

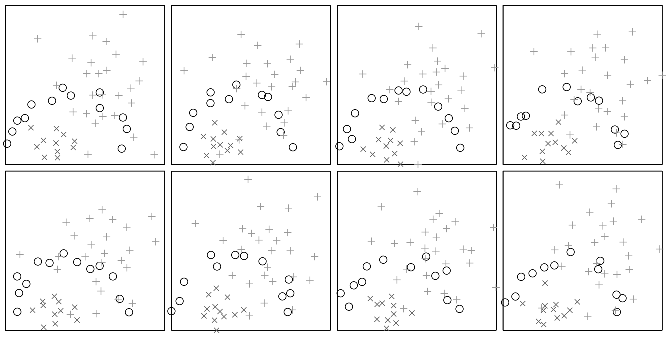


Fig. 2. Cone–torus – scatter plots of 8 DPS–S folds

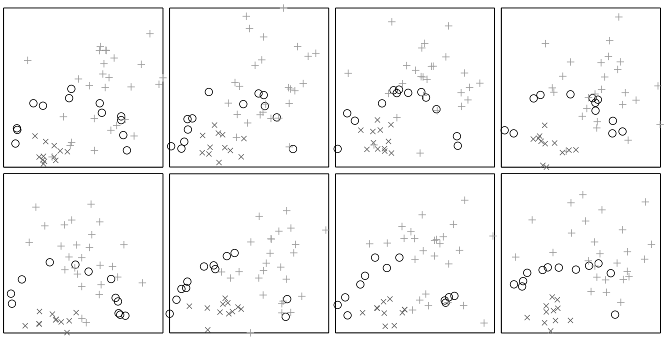


Fig. 3. Cone–torus – scatter plots of 8 CV folds

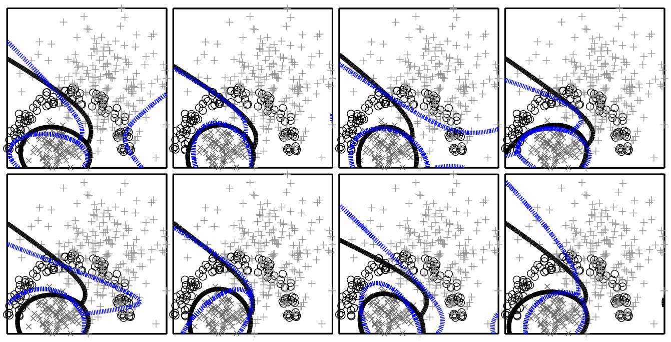


Fig. 4. Cone–torus – decision boundaries for *qdc* trained on DPS–S (solid line) and CV (dotted line) folds

the blue dotted line shows the boundaries for a single CV run. Notice, that for DPS the decision boundaries generally do not change their shape from one fold to another, as opposed to CV, where the boundaries are very unstable and change radically.

2) Synth–mat dataset: The Synth–mat dataset is a 2 dimensional mixture of 4 normal distributions and has been presented in Figure 1(b). Both classes have bimodal distribution – there are two Gaussians in each of them. The mean value of CiSI between all 8 folds and the original dataset is 0.75 for DPS and 0.66 for CV averaged over 10 runs ($\sigma = 0.12$). Since the scatter plots of all DPS and CV folds were already presented for the Cone–torus dataset and not much changes here, only the decision boundaries of a classifier trained as previously have been given in Figure 5. The boundaries appear stable for DPS and differ a lot from one CV fold to another.

B. Benchmark datasets

1) CiSI: Figure 6 presents the values of averaged CiSI between the original dataset and 8 folds generated using DPS and CV, for all 27 datasets used. Note, that although the index has been normalized to the $[0, 1]$ range, according to our earlier argument the values represent an ordinal scale. Also, for illustrative purposes, the Gaussian kernel width used for each dataset has been chosen to optimize correlation between bias and CiSI, as described in Section V-B5.

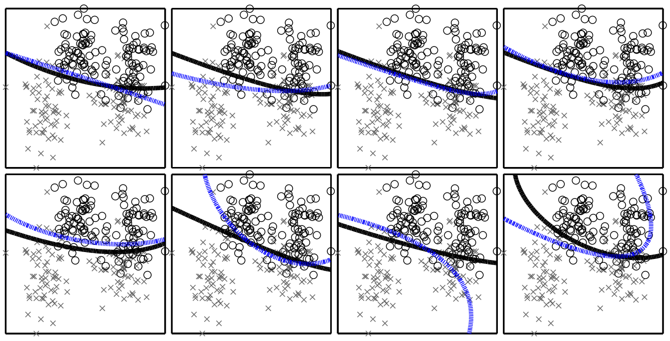


Fig. 5. Synth–mat – decision boundaries for *qdc* trained on DPS–S (solid line) and CV (dotted line) folds

The CiSI between the DPS folds and the original dataset is always higher than in the case of the CV folds, regardless of the number of folds (8 or 16). This is not surprising since the DPS splits have been obtained by maximization of CiSI. The picture is very similar for the between–fold CiSI given in Figure 7, where DPS is again an unquestionable leader.

2) Bias: The mean absolute bias for both DPS and CV can be seen in Figures 8 and 9. Due to off the chart bias values, the bootstrap estimator has not been shown here. The DPS approach has a bias comparable to the mean CV result, with slight advantage of the latter for roughly half of the datasets. Note however, that the DPS estimates are never as biased as the worst–case CV scenario, yet the result has been achieved with 10 times less computations.

A summary of the results, including the bootstrap estimator, can be found in Table III, where a mean value and standard deviation of bias (and variance) across all datasets and classifiers for each error estimation method has been given. Both DPS–U and DPS–S have on average the same bias with a tiny difference in its standard deviation. DPS–SU on the other hand comes very close to the repeated cross–validation, which is a result of combining both supervised and unsupervised methods. Note, that this combination does not require additional computations in order to obtain the splits, as all pairwise within–class distances form a subset of all pairwise distances for the whole dataset, which are calculated anyway by the unsupervised DPS. All DPS approaches also have mean bias and standard deviation lower than the worst–case CV scenario. The relatively high bias and variance values of the bootstrap estimator are the result of using just 8 bootstrap samples rather than recommended hundreds [11].

The performance advantage of DPS-SU over the remaining methods stems from a stabilizing and compensating effects similar to those of repeating CV. Figure 10 presents signed bias values for two classifiers chosen on the basis of their complexity (linear classifier and non-parametric classifier). As it can be seen in case of many datasets the signs of the DPS-U and DPS-S biases are opposite. It's also worth to note, that the behavior of both estimators is very consistent across all tested

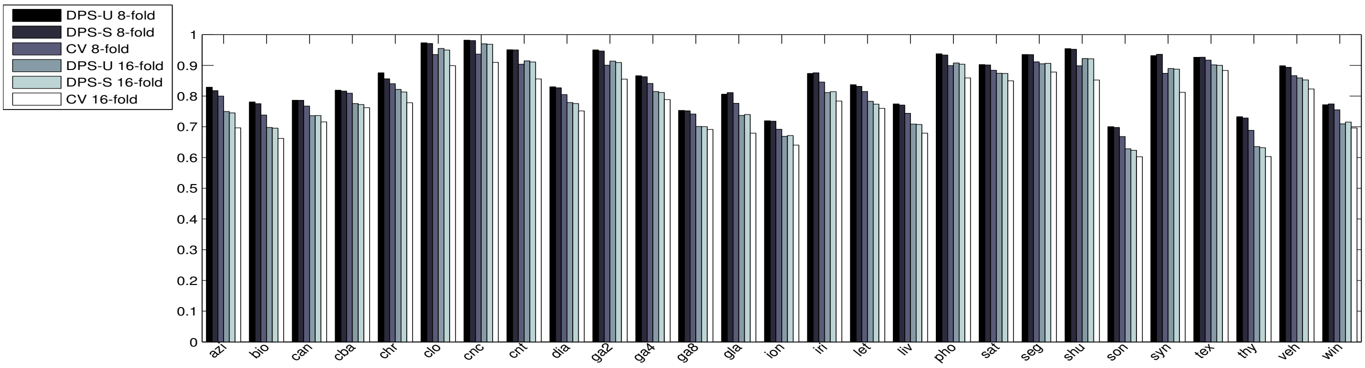


Fig. 6. Mean CiSI between each fold and the original dataset

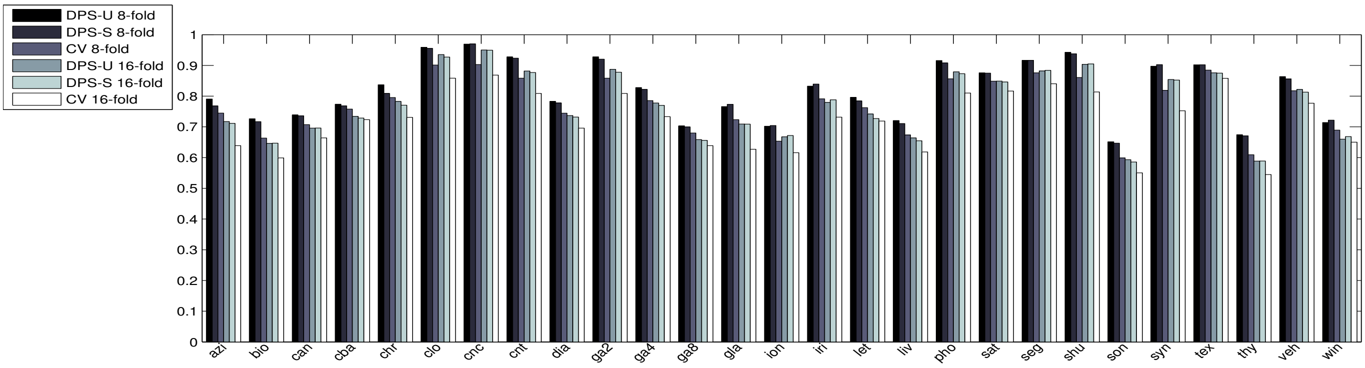


Fig. 7. Mean between–fold CiSI

TABLE III
BIAS AND VARIANCE SUMMARY FOR ALL DATASETS AND CLASSIFIERS

	DPS U	**DPS S**	**DPS SU**	**CV best**[a]	**CV mean**	**CV worst**	**Boot-strap**
BIAS-mean	0.028	0.028	0.028	0.024	0.027	0.033	0.150
BIAS-stdev	0.020	0.020	0.020	0.016	0.019	0.024	0.236
VAR-mean	0.060	0.050	0.039	0.052	0.063	0.074	0.026
VAR-stdev	0.033	0.033	0.023	0.030	0.035	0.040	0.015

[a] 'CV best' denotes the best cross–validation run out of 10 for each dataset/classifier pair in terms of lowest bias/variance. For CV the division of data which produced the lowest bias did not in general produce the lowest variance. Similar remarks apply to 'CV worst'.

classifiers in a sense, that the bias of one of them is almost always higher than the bias of the other for a given dataset.

3) Variance: The variance of error estimates can be seen in Figures 11 and Figure 12. Out of all three DPS approaches, once again DPS–SU demonstrates the best performance with average variance lower by 0.0130 than the best–case CV scenario (Table III), while DPS–S performs at the level of the best–case CV and DPS–U still outperforms 10 times repeated stratified cross–validation. Note, that both in terms of variance, DPS–S outperforms DPS–U and is additionally computationally cheaper (see Section IV-B). As a result good error estimation can be achieved with roughly 10% of computations required by 10 times repeated CV. For best results however one should resort to DPS–SU, which seems to stabilize the error estimates but requires more computational time.

4) Classifier selection: Selection of a single best model from a set is an important problem in machine learning. A typical selection criterion is the generalization error estimated using CV. The ranking of top 3 classifiers according to both DPS and 10x repeated CV for all datasets was given in Table IV. Note, that the overall ranking for all datasets is exactly the same for both error estimators, and reflects the ranks based on the true generalization error. Some differences are apparent when the results for each dataset are examined separately.

The last three rows in the table denote the number of datasets out of 27, for which the true top classifier was included in top 1, top 2 and top 3 classifiers according to each error estimation method. For CV, the best classifier has been correctly identified 19 times and has been included in the top 2 and top 3 classifiers 25 times. For the best DPS approach (DPS–SU) the numbers are similar – 20, 23 and 25, yet have been obtained at 20% of the computations required by CV.

The correlation coefficients for different error estimates and the true generalization error have been given in Table V. As shown, all tested error estimators are strongly correlated with the true error, and although there are some small differences, the correlation coefficient is never lower than 0.959.

5) Correlation between CiSI and bias: The ability to estimate the generalization error using a single DPS fold only would allow to reduce the computational cost of the estimation procedure by another order of magnitude, when compared to 10 times repeated CV. Figure 13 depicts the bias of the estimate calculated using a single DPS fold, which has been chosen on the basis of the lowest bias itself ('DPS–best'). Although in practice this kind of selection procedure is unfeasible, it shows that the method has some potential as for most datasets the bias is comparable with the one obtained using 10 times repeated cross–validation or even the best–case CV scenario. The problem however is how to choose the appropriate DPS fold. The value of CiSI seems to be an obvious choice. Note however, that there is no principled way of selecting the width σ of the Gaussian kernel for estimation

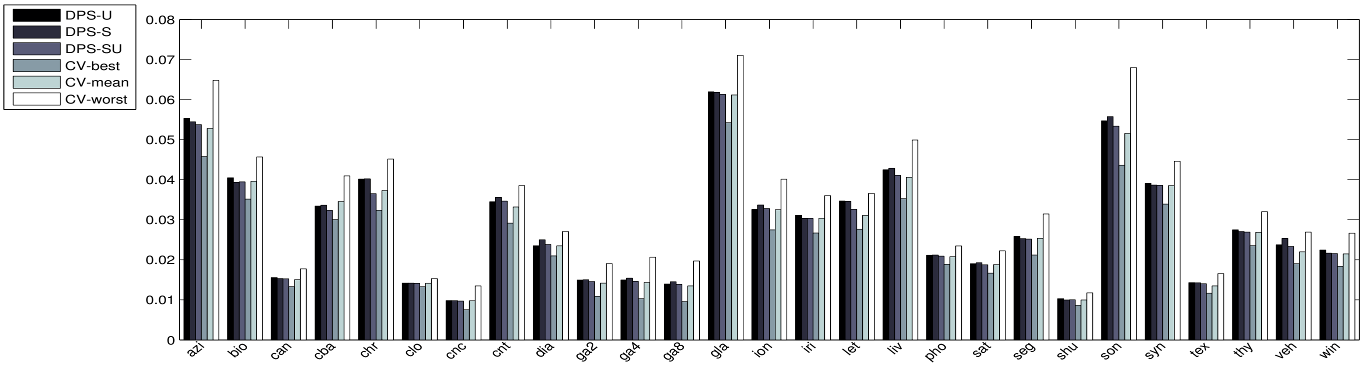


Fig. 8. Mean absolute bias (averaged over all classifiers)

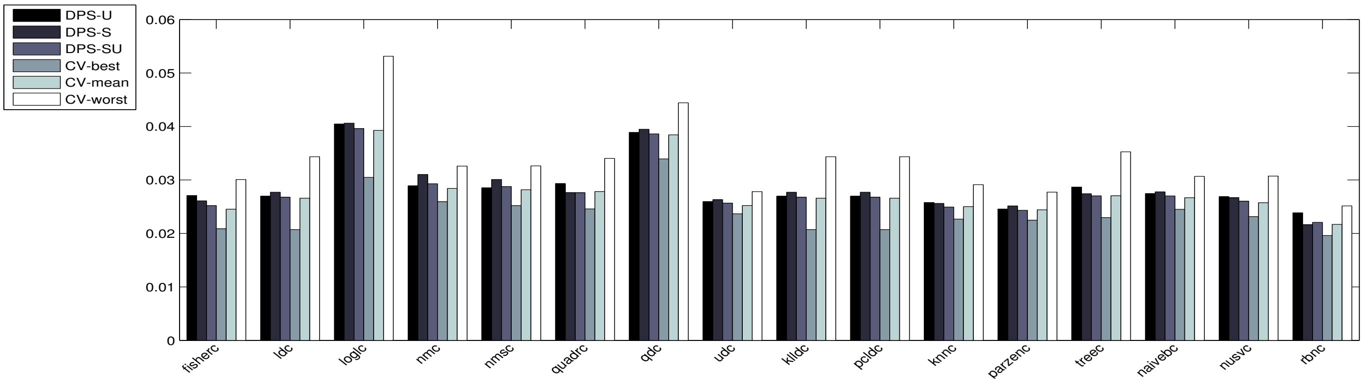


Fig. 9. Mean absolute bias (averaged over all datasets)

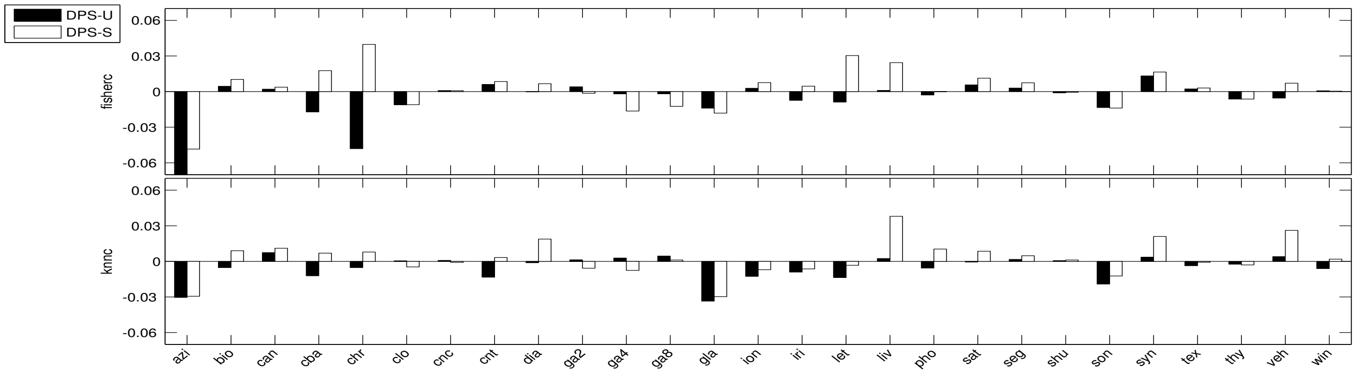


Fig. 10. Bias compensation of DPS-SU

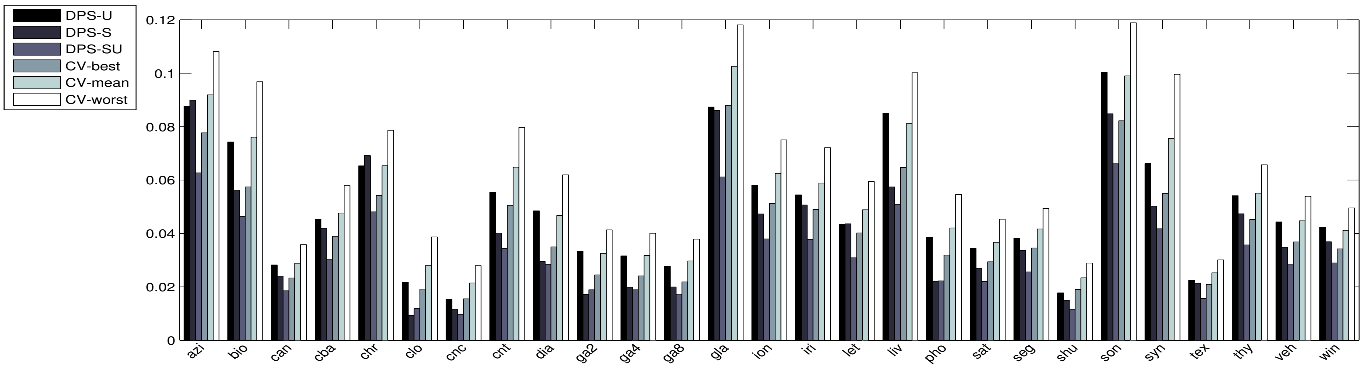


Fig. 11. Standard deviation of error estimate (averaged over all classifiers)

of CiSI and the estimated value can vary greatly depending on the choice of σ. We have therefore decided to evaluate the correlation between bias and CiSI. The experiment was performed for 8 and 16 DPS–S folds and the results can be seen in Figure 14. Note, that for the sake of calculating CiSI, σ was chosen using an exhaustive search in order to optimize the correlation. Thus the results given in Figure 14 represent the best–case scenario, for the optimal kernel width, which

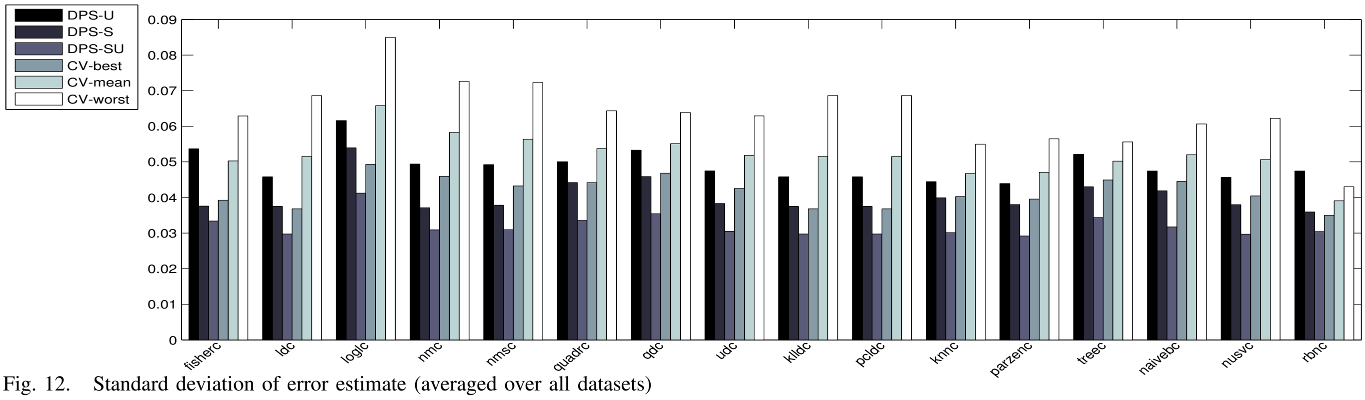


Fig. 12. Standard deviation of error estimate (averaged over all datasets)

TABLE IV
RANKING OF TOP 3 CLASSIFIERS

dataset	true	DPS–U	DPS–S	DPS–SU	CV mean
azi	11 14 12	11 12 2	11 12 14	11 12 2	11 12 14
bio	6 7 8	6 7 8	6 7 8	6 7 8	6 7 8
can	12 2 9	12 2 9	15 12 2	15 12 2	12 15 2
cba	12 2 9	12 2 9	12 2 9	12 2 9	12 2 9
chr	14 8 11	14 8 11	14 8 2	14 8 11	14 8 11
clo	15 1 2	1 2 3	15 1 2	15 1 2	15 1 2
cnc	1 15 3	1 15 2	1 15 5	1 15 5	1 15 5
cnt	16 12 13	16 12 7	12 16 13	16 12 13	16 12 13
dia	1 3 2	2 9 10	1 11 2	1 2 9	1 3 2
ga2	1 2 3	15 4 2	2 3 9	2 3 9	3 1 2
ga4	1 2 3	1 4 2	15 5 2	15 1 2	15 4 3
ga8	16 3 1	16 1 5	16 5 4	16 5 1	16 15 5
gla	3 15 2	2 9 10	2 9 10	2 9 10	2 9 10
ion	14 11 7	14 7 11	14 13 7	14 13 7	14 7 11
iri	2 9 10	2 9 10	2 7 9	2 9 10	2 9 10
let	11 12 7	12 11 2	12 11 2	12 11 2	12 11 2
liv	1 3 2	1 3 2	11 1 3	1 3 11	1 3 11
pho	11 16 12	16 12 11	11 12 16	11 16 12	11 12 16
sat	11 12 2	11 12 14	11 12 2	11 12 2	11 12 7
seg	11 3 2	3 11 2	11 2 9	11 3 2	3 11 2
shu	13 1 16	13 16 1	13 1 16	13 16 1	13 1 16
son	12 11 15	12 11 14	12 11 14	12 11 14	12 11 2
syn	14 12 16	14 12 16	12 14 16	14 12 16	12 14 16
tex	2 9 10	2 9 10	2 9 10	2 9 10	2 9 10
thy	8 15 7	15 6 7	7 8 11	15 7 8	15 8 11
veh	7 6 3	6 7 2	7 6 2	7 6 2	7 6 2
win	7 6 2	6 1 4	4 6 7	6 4 7	6 7 4
overall	2 9 10	2 9 10	2 9 10	2 9 10	2 9 10
in top 1	27/27	17/27	17/27	20/27	19/27
in top 2	27/27	20/27	23/27	23/27	25/27
in top 3	27/27	21/27	24/27	25/27	25/27

TABLE V
CORRELATION BETWEEN ‘TRUE’ ERROR AND ITS ESTIMATES

correlation	DPS–U	DPS–S	DPS–SU	CV
per dataset (8 folds)	0.9594	0.9657	0.9676	0.9710
per classifier (8 folds)	0.9967	0.9975	0.9976	0.9973
per dataset (16 folds)	0.9640	0.9646	0.9671	0.9695
per classifier (16 folds)	0.9964	0.9969	0.9969	0.9969

in practice is not known a priori. The correlation varies from about -0.1 to -0.6 depending on the dataset. The bias of an estimate obtained using a single DPS fold chosen on the basis of highest CiSI value is always higher even than the worst–case CV scenario bias (‘DPS–optim’ in Figure 13). The CiSI is only slightly to moderately correlated with the bias, even for an optimal choice Gaussian kernel width and hence cannot be used to select a single best fold which minimize the bias.

6) Combining classifiers: In this experiment a simple ensemble model based on the majority voting rule was built. It is believed, that the classifiers used in a combination should be diverse, which enforces complementarity of the ensemble members [35]. One way to enforce diversity is cross–training, a technique based on cross–validation, which combines all models obtained during a single or repeated CV run. For this experiment the two synthetic datasets from Section V-A have been used. Both datasets were split into 8 folds using DPS–S and CV and then for each classifier listed in Table II an ensemble model was built by combining 8 models trained on all but one fold in turn, using the majority voting rule. For CV this procedure has been repeated 10 times. Each combination was then tested on an independent test set. In order to monitor performance of the combinations, a single control model trained using all 8 folds was also used.

The results have been depicted in Figures 15 and 16. In most cases, combinations based on DPS folds do not improve on the performance of a single control model. This was expected, as for each classifier all ensemble members should be very similar, since they were all trained using representative data subsets. For the combinations based on CV, some improvement can be observed even in the worst case scenario.

To illustrate this issue, discrete error distribution plots showing the probability of various numbers of ensemble members being in error for the same input instance, have been given in Figures 17 and 18. The classifiers used to produce these plots (*qdc* and *treec*) have been chosen for illustrative purposes. The area of the shaded region in each figure represents the error of the combined model (the ties, i.e. when there were exactly 4 votes for each of the two classes in the case of the Synth-mat dataset, were resolved randomly). Note, that for DPS most of the mass is concentrated in the corners of the plots, meaning unanimous classification decisions in most cases and proving that the classifiers are indeed very similar.

In case of CV the situation is different. In Figure 17 some mass is scattered all over the plot, meaning that sometimes the classifiers tend to disagree, demonstrating complementarity. As it can be seen, in this case the stochastic nature of CV positively affects the performance by introducing diversity to the ensemble. This result also confirms, that if the goal is to select a single best model, it is much safer to use DPS, minimizing the risk of choosing a bad model due to the discussed stability of decision boundaries. In the context of ensemble models however, this feature of DPS becomes a disadvantage and it's usually better to use a stochastic method.

VI. DISCUSSION

The presented Density Preserving Sampling procedure is an attractive alternative for cross–validation. For the purpose of

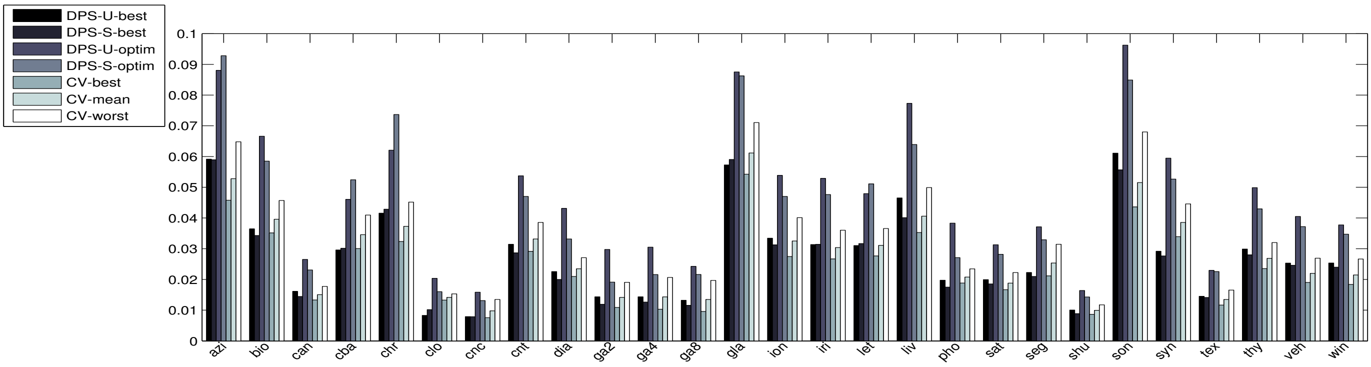


Fig. 13. Bias of DPS error estimate calculated using a single fold (averaged over all classifiers)

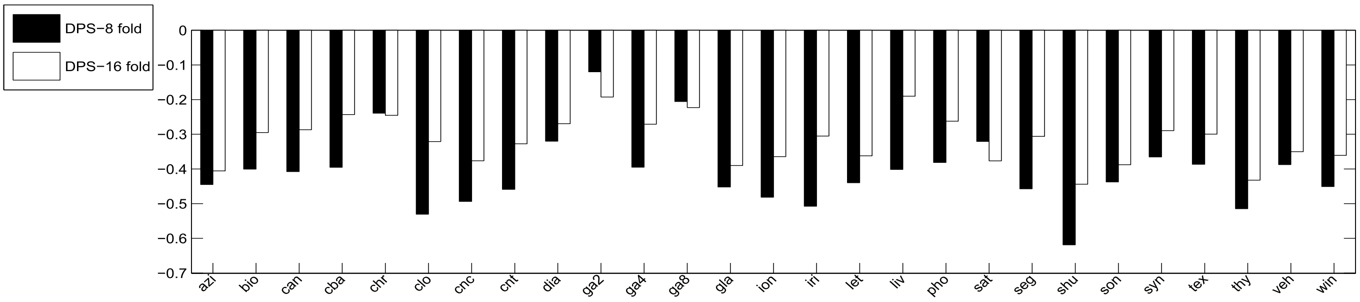


Fig. 14. Correlation between bias and CiSI

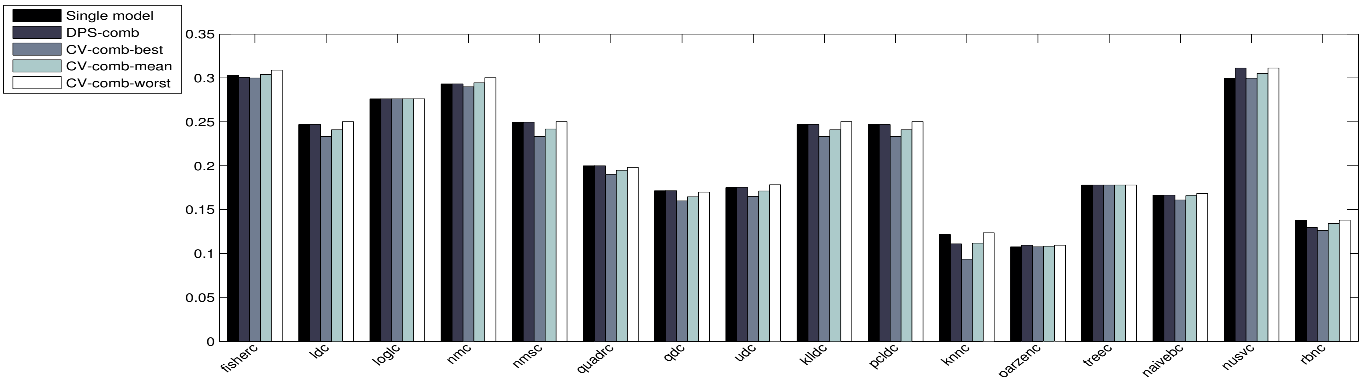


Fig. 15. Single model v. combination errors for Cone–torus dataset

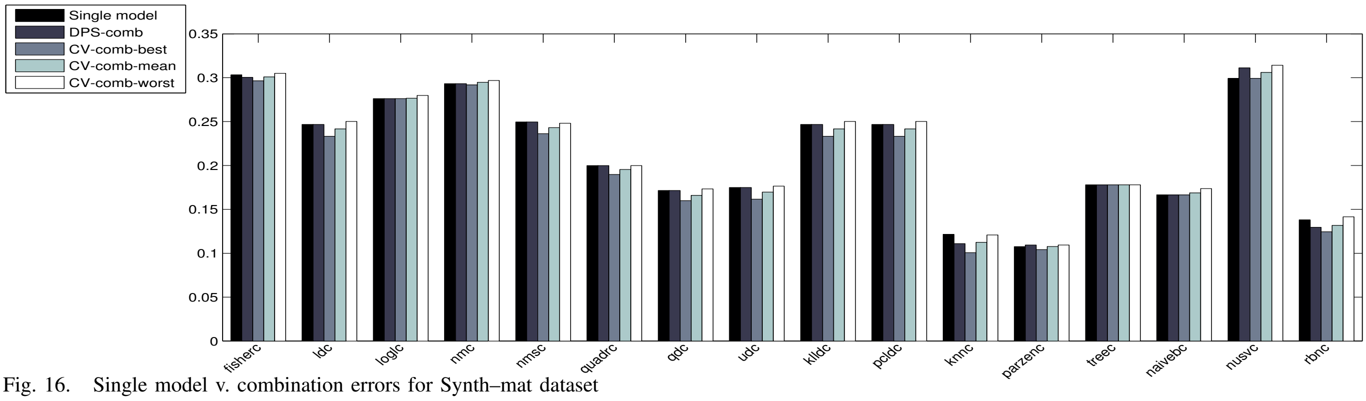
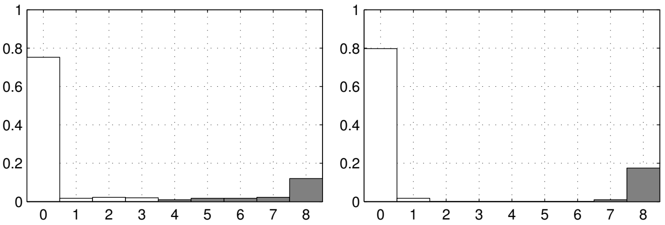


Fig. 16. Single model v. combination errors for Synth–mat dataset

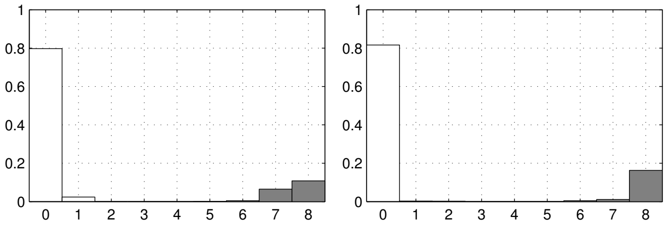
the generalization error estimation, k–fold CV is without a doubt the most widely and commonly used technique, due to its universal character, simplicity and effectiveness. Its stochastic nature however requires the estimation to be repeated multiple times for different random divisions of the data, in order to circumvent the risk of obtaining the best/worst–case scenario estimate, which as demonstrated in this paper can be highly biased and can have a large variance. The need for running the procedure multiple times makes it computationally expensive, forcing the researchers to seek compromise elsewhere, e.g. by not calculating the full gradient during optimization or taking other shortcuts, leading to suboptimal solutions. The proposed



(a) CV–best (0.1616)

(b) DPS (0.1715)

Fig. 17. Discrete Error Distributions for Cone–torus dataset and *qdc* (number of classifiers on X-axis, probability on Y-axis, error rates given in brackets)



(a) CV–best (0.1790)

(b) DPS (0.1790)

Fig. 18. Discrete Error Distributions for Synth–mat dataset and *treec* (number of classifiers on X-axis, probability on Y-axis, error rates given in brackets)

DPS procedure is however deterministic; it does not need to be repeated to improve the quality of the error estimate, at the same time producing results comparable to repeated CV when it comes to bias, and superior to CV in terms of the variance of obtained estimates, at 5–10x lower computational cost.

Another related application area of CV is parameter estimation. Since for some models the objective function is not differentiable wrt. all its parameters, the optimization procedure must resort to a search in the parameter space. One example of such situation is the *k*–NN classifier, for which the number of nearest neighbours *k* is usually being set by testing a number of possible values using CV. In such case, as the search itself might be very costly depending on the dimensionality of the search space, the cross–validation is usually not being repeated in order to save computations. As before, due to the non–deterministic nature of CV, this can lead to suboptimal decisions based on highly biased performance estimates (worst–case scenario). Note, that it also applies to other algorithms requiring calculation of performance estimates repeated many times like e.g. feature selection. The benefit of using DPS rather than CV in these scenarios can be tremendous.

In case of some machine learning methods it is a common practice to cross–train multiple models and select the best performing one. The cross–training procedure is analogous to CV, with the difference that the obtained models instead of being discarded, are considered as candidates for a final solution. This applies especially to models like decision trees, which cannot be retrained using the full dataset due to their instability. The danger here is the combination of a relatively unstable error estimator with an unstable learning method, which may lead to selection of one of the worst models rather than the best. On the other hand, models trained using various DPS splits will likely be much more similar to each other, minimizing the risk and cost of incorrect choice.

The final example of possible application of random sampling procedures is early stopping, a technique widely used in training of universal approximators to prevent overfitting. In this approach a randomly selected subset of the data is used for continuous monitoring of model performance during training, in order to stop it when the validation error starts to increase, signalling overfitting. The risk of using unrepresentative validation set is obvious. Although the behavior of DPS in conjunction with early stopping has not been addressed in this paper, it forms an interesting and promising research direction.

It is worth to note that a similar attempt to promote usage of techniques alternative to standard CV had already taken place in the past. In [36] the authors propose the Distribution–Balanced Stratified Cross–Validation method, similar in principle to DPS-S. The weakness of this method is however an arbitrary choice of the reference point for ordering of instances for subsequent sampling. If the reference point is chosen poorly, this can result in areas of the input space not being represented in the sampled subsets, although at the same time it allows to perform the sampling in time linear in the size of the dataset. The experimental part of [36] includes only 9 datasets and one classifier, which is one of the reasons for the lack of wider recognition of the method, although it has recently been investigated more extensively in [37].

VII. Conclusions

The correntropy–inspired density–preserving data sampling procedure developed and investigated in this paper is an alternative for cross–validation in many applications. Unlike CV, DPS is deterministic, which eliminates the need for multiple repetitions of the sampling procedure to obtain reliable results, considerably reducing the computational burden.

The main property of the proposed method is that it aims to produce only representative splits, which has many implications outlined in previous sections. The experiments performed on a comprehensive set of public benchmark datasets and a number of standard classifiers have revealed that:

- For generalization error estimation, DPS is slightly more biased than 10x repeated cross–validation but has much lower variance, often lower than the best–case CV scenario. The DPS bias in all cases is also much lower than in the worst–case CV scenario.
- The decision boundaries of a classifier trained on DPS folds are much more stable than in the case of a CV folds, which is the result of representativeness of the subsets generated by DPS. The stability of models trained on various DPS divisions of the dataset has been confirmed in experiments involving ensemble models.
- For model ranking and selection, DPS is at least as good as 10x repeated CV, at much lower computational cost.

We believe that the one of the strengths of correntropy stems from its connection to the Information Theoretic Learning framework. In ITL data vectors are modelled as particles, whose local interactions determine the global behavior of the whole system. Hence future research will focus on discovery of other ITL-based objective functions for selection of a single representative fold to be used for error estimation, effectively reducing the computational requirements by another order of magnitude. This approach appears even more valid and viable in the light of the empirical results given in [20], which show that one of the most intuitive choices for representative sampling – the PDF divergence measures – is not feasible due to difficulties with their estimation. Another interesting future research direction is a version of DPS applicable to streaming data in the presence of concept drift as well as an alternative, more computationally efficient CiSI optimization scheme to enable application of DPS to large datasets.

Acknowledgements

The research leading to these results has received funding from the European Union 7^{th} Framework Programme (FP7/2007-2013) under grant agreement 251617.

References

[1] R. Duda, P. Hart, and D. Stork, *Pattern Classification 2nd ed.* New York, USA: John Wiley & Sons, 2001.

[2] T. Cover, “Learning in Pattern Recognition.” Stanford University CA, Stanford Electronics Labs, Tech. Rep., 1968.

[3] B. Efron, “Bootstrap methods: another look at the jackknife,” *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[4] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 2, no. 12. Morgan Kaufmann, 1995, pp. 1137–1145.

[5] A. Harzing, “Publish or Perish,” www.harzing.com/pop.htm.

[6] D. Anguita, A. Ghio, L. Oneto, and S. Ridella, “In-sample and out-of-sample model selection and error estimation for support vector machines,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 9, pp. 1390–1406, 2012.

[7] W. Liu, P. Pokharel, and J. Principe, “Correntropy: A Localized Similarity Measure,” in *Proceedings of the International Joint Conference on Neural Networks*, 2006, pp. 4919–4924.

[8] M. Budka and B. Gabrys, “Correntropy–based density–preserving data sampling as an alternative to standard cross–validation,” in *Proceedings of the IEEE World Congress on Computational Intelligence*. IEEE, 2010, pp. 1437–1444.

[9] A. Antos, L. Devroye, and L. Gyorfi, “Lower bounds for Bayes error estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 7, pp. 643–645, 1999.

[10] A. Jain, R. Duin, and J. Mao, “Statistical pattern recognition: A review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[11] S. Weiss and C. Kulikowski, *Computer systems that learn*. San Francisco, USA: Morgan Kaufmann, 1991.

[12] B. Efron and R. Tibshirani, *An introduction to the bootstrap*. Chapman & Hall/CRC, 1993, vol. 57.

[13] A. Molinaro, R. Simon, and R. Pfeiffer, “Prediction error estimation: a comparison of resampling methods,” *Bioinformatics*, vol. 21, no. 15, pp. 3301–3307, 2005.

[14] J. Kim, “Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap,” *Computational Statistics & Data Analysis*, vol. 53, no. 11, pp. 3735–3745, 2009.

[15] S. Borra and A. Di Ciaccio, “Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods,” *Computational statistics & data analysis*, vol. 54, no. 12, pp. 2976–2989, 2010.

[16] Y. Dodge, D. Cox, D. Commenges, A. Davison, and P. Solomon, *The Oxford dictionary of statistical terms*. Oxford, UK: Oxford University Press, 2006.

[17] A. Cichocki and S. Amari, “Families of Alpha–Beta–and Gamma–Divergences: Flexible and Robust Measures of Similarities,” *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[18] S. Kullback and R. Leibler, “On information and sufficiency,” *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[19] J. Lin, “Divergence measures based on the Shannon entropy,” *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[20] M. Budka, B. Gabrys, and K. Musial, “On accuracy of PDF divergence estimators and their applicability to representative data sampling,” *Entropy*, vol. 13, no. 7, pp. 1229–1266, 2011.

[21] M. Budka, “Physically inspired methods and development of datadriven predictive systems,” Ph.D. dissertation, PhD thesis, Bournemouth University, UK, 2010, 2010.

[22] J. Principe, D. Xu, and J. Fisher, “Information Theoretic Learning,” in *Unsupervised Adaptive Filtering*, S. Haykin, Ed. John Wiley & Sons, 2000, pp. 265–319.

[23] J. Principe, D. Xu, Q. Zhao, and J. Fisher, “Learning from Examples with Information Theoretic Criteria,” *The Journal of VLSI Signal Processing*, vol. 26, no. 1, pp. 61–77, 2000.

[24] I. Santamaría, P. Pokharel, and J. Principe, “Generalized correlation function,” *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2187–2197, 2006.

[25] ——, “Generalized correlation function: Definition, properties, and application to blind equalization,” *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2187–2197, 2006.

[26] W. Liu, P. Pokharel, and J. Principe, “Correntropy: properties and applications in non–Gaussian signal processing,” *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.

[27] ——, “Error Entropy, Correntropy and M–Estimation,” in *Proceedings of the 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*. IEEE, 2006, pp. 179–184.

[28] M. Budka and B. Gabrys, “Ridge regression ensemble for toxicity prediction,” *Procedia Computer Science*, vol. 1, no. 1, pp. 193–201, 2010.

[29] J. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[30] A. Asuncion and D. Newman, “UCI Machine Learning Repository,” 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[31] “Enhanced Learning for Evolutive Neural Architecture (ELENA) Database,” 1995. [Online]. Available: http://www.dice.ucl.ac.be/mlg/?page=Elena

[32] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, “PR–Tools 4.1, A MATLAB Toolbox for Pattern Recognition,” 2007, http://prtools.org.

[33] L. Kuncheva, *Fuzzy classifier design*. Heidelberg, Germany: Physica Verlag, 2000.

[34] B. Ripley, *Pattern recognition and neural networks*. Cambridge, UK: Cambridge University Press, 1996.

[35] L. Kuncheva, *Combining pattern classifiers: methods and algorithms*. New York, USA: John Wiley & Sons, 2004.

[36] X. Zeng and T. Martinez, “Distribution-balanced stratified cross-validation for accuracy estimation,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 1, pp. 1–12, 2000.

[37] J. Moreno-Torres, J. Sáez, and F. Herrera, “Study on the impact of partition-induced dataset shift on¡ formula formulatype=,” *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1304–1312, 2012.

Marcin Budka received a dual BA+MA degree in finance and banking from the University of Economics, Katowice, Poland, in 2003, BSc in computer science from University of Silesia, Poland, in 2005 and PhD in computational intelligence from Bournemouth University, UK, in 2010, where he currently holds the Lecturer in Computational Intelligence position. His current research interests include Information Theoretic Learning, meta-learning, adaptive systems, ensemble models and complex networked systems with focus on evolution and dynamics of social networks.

Bogdan Gabrys received MSc in electronics and telecommunication from Silesian Technical University, Poland, in 1994, and PhD in computer science from Nottingham Trent University, UK, in 1998. After many years of working at different Universities, he moved to the Bournemouth University in January 2003, where he holds a Chair in Computational Intelligence position and acts as a director of the Smart Technology Research Centre within the School of Design, Engineering and Computing. His current research interests lay in a broad area of intelligent, complex adaptive systems and include a wide range of machine learning, biologically/nature inspired learning and hybrid intelligent techniques encompassing data and information fusion, learning and adaptation methods, multiple classifier and prediction systems, processing and modelling of uncertainty in pattern recognition, diagnostic analysis and decision support systems.