Correntropy–based density–preserving data sampling as an alternative to standard cross–validation

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

***Abstract*— Estimation of the generalization ability of a predictive 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–based 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, which has been investigated using a set of publicly available benchmark datasets and standard classifiers.**

I. Introduction

Estimation of the generalization ability of a predictive model is an important issue in the machine learning field, especially that it is independent of the actual model used. Generalization accuracy estimates are not only used as indicators of the expected performance of the developed classifier or regressor on previously unseen data, but are also commonly used for model ranking and selection [7].

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 [5] dating back to 1968, and bootstrap [9] developed in the 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 [12], and currently estimated to have more than 1500 citations[1].

The basic idea, which is being 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. The main difference between various techniques is the way the generalization error is being calculated, the size of the subset reserved for testing or whether the procedure is repeated multiple times or not. They have however something in common, and that is the way in which the testing subset is being 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. In order to address this issue, stratified sampling approaches have been developed [12], which try to increase the representativeness at the expense of independence, and are able to achieve better results than their non–stratified counterparts.

Inspired by the success of stratified sampling approaches, in this paper we propose a density preserving sampling procedure (DPS), which further sacrifices the independence of the test set to enforce its representativeness. The method achieves this goal by optimizing the correntropy, a recently developed, non–parametric similarity measure of the probability density functions [14], and as shown later produces accurate generalization estimates requiring a fraction of computations when compared to cross–validation.

The remainder of this paper is organized as follows. In Section II the problem of estimation of generalization error is introduced, together with standard estimation techniques and criteria of their evaluation. Section III describes the concept of Information Theoretic Learning including online manipulation of Renyi's quadratic entropy and the definition of correntropy. In Section IV, the novel Density Preserving Sampling procedure is derived. The experimental results are given in Section V, while the discussion and conclusions can be found in Sections VI and VII respectively.

II. Estimation of the generalization error

Generalization error is the error a predictive model will make on novel, previously unseen data, generated from the same distribution as the data used to develop the model [7]. Low generalization error is thus a sign of a good match between the model and the problem and no overfitting [4].

In general it is impossible to obtain a closed form solution for calculation of the generalization error or even for calculation of tight bounds for the error, in all but the simplest cases [2]. 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

Marcin Budka and Bogdan Gabrys are with the Computational Intelligence Research Group, Bournemouth University, School of Design, Engineering and Computing, Poole House, Talbot Campus, Fern Barrow, Poole BH12 5BB, United Kingdom (phone: +44 1202 524111 ext. 61463; email: mbudka@bournemouth.ac.uk or marcin@budka.co.uk / bgabrys@bournemouth.ac.uk).

[1]According to Harzing's Publish or Perish citation retrieval system (http://www.harzing.com/pop.htm), which uses the Google Scholar database (http://scholar.google.com/).

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 being 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 [21], 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 obvious drawback of the hold–out method is that unless both datasets are large enough (which is a vague term by itself), different estimates will be obtained from one run to another. A workaround, known as random subsampling [10], repeats the hold–out procedure multiple times and averages the results. This procedure 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 [12]. In order to circumvent these issues, more advanced resampling techniques have been developed. Yet, the hold–out method is still being used when dealing with large datasets, as in this case other techniques quickly become untractable. It is also assumed that more advanced resampling techniques are simply not needed for large amounts of data.

B. Cross–validation

Cross–validation is a widely used standard statistical technique for estimation of model generalization ability, applied with a great success to both classification and regression problems [4], [7]. In k–fold cross–validation the whole available dataset is first 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 $k-1$ subsets and tested using the validation subset. The estimate of the generalization error is then calculated as a mean value of all validation errors, while the standard deviation of the validation error can be used to approximate the confidence intervals of obtained error estimate. The whole procedure thus requires development of exactly k models. Since the results obtained in the setting described above 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 frequently 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 error estimates but with high variance 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 [21].

In order to improve the accuracy of the estimates obtained, a stratified cross–validation approach is used in practice, which samples the data in a way that approximates the percentage of each class in every fold [12]. For regression problems, stratified cross–validation produces folds with equal mean values of the target variable [10].

C. Bootstrap

Bootstrap is a second commonly used generalization error estimation procedure [4], [7], especially useful when dealing with small datasets [21]. 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 picked during the sampling procedure form the test set. The probability of each instance ending up in the test set is thus:

$$\left(1-\frac{1}{m}\right)^{m} \approx e^{-1} \approx 0.368 \tag{1}$$

Due to the fact that the probability of each instance being picked for training is $1-0.368 = 0.632$, the method is also often called the '0.632 bootstrap' [21]. Since the error estimate obtained using test data only would be overly pessimistic (only about 63.2% of instances are used for training every time), the generalization error estimate is calculated using weighted test and training errors. In general, the more times the whole process is repeated, the more accurate the estimate. A comprehensive comparative study of cross–validation and bootstrap can be found in [12].

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 [12]. 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 error estimation procedure. There is another important parameter – the variance, which measures the variability of the 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 [12]. A good 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 [4].

III. Information Theoretic Learning (ITL)

Information Theoretic Learning is a procedure of adapting the parameters of a learning machine using information theoretic criterion [17]. The goal of learning can be stated as exploration and exploitation of redundancies from a single or multiple sources of information the learning machine is exposed to. This shifts the problem towards quantification and manipulation of redundancy, making Shannon's information theory the ultimate framework of machine learning [18].

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 probability density functions given in an analytic form [18]. Although it is possible to use numerical approximation, it quickly becomes intractable as the dimensionality of the input space grows [17]. Information Theoretic Learning framework should thus allow for both non–parametric estimation and manipulation of entropy and various divergence measures, enabling training of both linear and nonlinear mappers by transferring as much information as possible from the training data into parameters of the system [18].

A. Renyi's quadratic entropy

Entropy is a measure of the uncertainty associated with a random variable. It quantifies the average information content that is missing due to the unknown value of the variable and is the main criterion used in ITL approaches.

Denoting by $p(y)$ the probability density of the random variable y, Shannon's differential entropy is given by:

$$H_S(y) = \int p(y) \log \frac{1}{p(y)}\, dy = -\int p(y) \log p(y)\, dy \quad (2)$$

Calculation of Shannon's entropy usually requires the density function to be given in an analytic form. There are however other definitions of entropy that can be used in the ITL framework. One of them is Renyi's entropy of order α, which for a continuous random variable is given by:

$$H_{R_\alpha}(y) = \frac{1}{1-\alpha} \log \int p(y)^\alpha\, dy \quad (3)$$

Renyi's entropy involves calculation of the integral of the power of PDF rather than integral of the logarithm as in the case of Shannon's counterpart, which is much easier to estimate [18]. Moreover, Shannon's entropy is the limiting case of Renyi's entropy when $\alpha \to 1$. For practical applications the choice of $\alpha = 2$ is a good compromise between robustness and computational complexity ($O(n^2)$) [18], leading to the definition of Renyi's quadratic entropy:

$$H_{R_2} = -\log \int p(y)^2\, dy \quad (4)$$

The most important property of Renyi's entropy from the point of view of ITL is that the extrema of H_S and H_R overlap [17], so both definitions are equivalent for the purpose of entropy optimization.

The above criterion is however still useless without a good estimate of the probability density function, which fortunately can be obtained and efficiently integrated into (4) by using the Parzen window density estimator [7]. Denoting by $G(y,\ \sigma^2 I)$ a spherical Gaussian kernel centered at y with diagonal covariance matrix, the PDF can be estimated as:

$$p(y) = \frac{1}{N} \sum_{i=1}^{N} G(y - y_i,\ \sigma^2 I) \quad (5)$$

Substituting 5 into 4 and using the convolution property of the Gaussian kernel yields:

$$H_{R_2}(y) = -\log \int p(y)^2\, dy = -\log V(y) \quad (6)$$

$$V(y) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(y_i - y_j,\ 2\sigma^2 I) \quad (7)$$

Renyi's entropy of order α calculates the interactions between α–tuplets of samples, so the higher the value of α, the more information about the structure of the dataset can be extracted [18] but the computational complexity – $O(n^\alpha)$ – quickly becomes prohibitive.

If some imaginary particles were placed on top of each data sample a potential field would be created, since $G(y_i - y_j,\ 2\sigma^2 I)$ is always positive and decays exponentially with the square of the distance between y_i and y_j [17]. The samples can thus be referred to as Information Particles while $V(y)$, which is an averaged sum of all pairs of interactions and represents the total potential energy of the dataset – Information Potential. By analogy to classical physics the gradient of potential energy is a force, which wants to drag the particles to a state with minimum potential. This behaviour can be immediately used for training of adaptive systems with forces taking the place of the injected error and used for adjusting parameters of the model [18].

B. Auto– and cross–correntropy

A Generalized Correlation Function (GCF) for a stochastic process x_t has been defined in [20] as:

$$V(t_1, t_2) = E[\phi(x_{t_1}), \phi(x_{t_2})] = E[k(x_{t_1}, x_{t_2})] \quad (8)$$

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. It has been proven, that 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 [20]. The function has been named auto–correntropy and is a preferred choice over traditional methods also due to taking advantage of all even moments of the PDF.

The idea of auto–correntropy has been further developed in [14] to support a general case of two arbitrary random variables. The new measure, named cross–correntropy (or correntropy) is defined for random variables X and Y as:

$$V(X, Y) = E[\phi(X), \phi(Y)] = E[k(X, Y)] \quad (9)$$

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. As a result, correntropy can also be defined as the integral of the joint probability density along the line $x = y$:

$$V(X, Y) \approx \int p(x, y)\ |_{x=y=u}\, du \quad (10)$$

The joint PDF can be estimated from the data using the Parzen window method:

$$p(x,y) \approx \frac{1}{N}\sum_{i=1}^{N} G(x-x_i,\sigma^2 I)G(y-y_i,\sigma^2 I) \qquad (11)$$

By integrating above equation along the $x=y$ line and once again using the convolution property of Gaussian functions, the estimate of correntropy is finally obtained as:

$$V(X,Y) \approx \frac{1}{N}\sum_{i=1}^{N} G(x_i-y_i,2\sigma^2 I) \qquad (12)$$

The correntropy can thus 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 [14], [16]. 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 7 and named Cross Information Potential [16]. The correntropy has been successfully employed as a localized, outlier–resistant similarity measure for supervised learning [15], [16], [11].

IV. Density Preserving Sampling procedure

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 run 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. For this reason, 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 will converge to the true error values. For $k-$fold cross–validation using $m-$element dataset this could mean averaging over all $\binom{m}{m/k}$ possibilities of choosing m/k instances out of m (the so called 'complete cross–validation' [12]), 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 [6]. Since these characteristics are reflected by the probability density function, the more similar the distribution of the sample to the distribution of the population, the more representative this sample is. The correntropy described in section III-B can be used to measure the similarity between two distributions and thus to measure the 'representativeness' of the sample. Moreover, it is also 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.

A. Correntropy for unsorted sets with unequal cardinalities

Equation 12 defines 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 correntropy to be possible: the datasets (1) must be ordered, (2) must have the same dimensionality and (3) must have the same number of objects. While the second requirement is irrelevant for sampling, as each subset of objects necessarily has the same dimensionality as the set from which it originates, the remaining two requirements may pose a problem.

For some applications like e.g. supervised learning, all the above requirements are met automatically – if X denotes the output of a mapper and Y denotes the target value, $|X|=|Y|$ and x_i is the prediction of y_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. Moreover, the datasets may have different cardinalities e.g. when one wants to calculate the correntropy between the original dataset and its subset.

To address the ordering issue the following approach is adapted. For every instance $x_i, i \in (1..N)$, the Gaussian is centred at (x_i-y_j), such that:

$$j = \underset{j}{\operatorname{argmin}} \parallel x_i - y_j \parallel, j \in (1..N) \qquad (13)$$

where $\parallel \cdot \parallel$ denotes the Euclidean norm. In other words y_j is selected to be as close to x_i as possible. Both x_i and y_j are then removed from their respective sets and the procedure is repeated until all instances are exhausted. The generalized, instance ordering insensitive formula for calculation of correntropy thus becomes:

$$V(X,Y) \approx \frac{1}{N}\sum_{i=1}^{N} G(x_i-y_j,2\sigma^2 I)$$
$$j = \underset{j}{\operatorname{argmin}} \parallel x_i - y_j \parallel, \quad j \in J_{avail} \qquad (14)$$

where the set J_{avail} contains all 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, that is without loss of generality if $N_X > N_Y$, the approach outlined above 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. Correntropy is then calculated between X and Y_N and the calculation will terminate after exactly N_X steps.

For the correntropy values to be more comparable for different experiments, we scale $V(X,Y)$ to fit into the $0..1$ range, by dividing each $G(x_i-y_j,2\sigma^2 I)$ in the sum in Equation 14 by $G(0,2\sigma^2 I)$. Every correntropy value given in the rest of this paper has been scaled. Note however, that the correntropies should be compared with caution as their absolute difference can be made almost arbitrarily large by manipulating the Parzen window width parameter σ. For this reason the correntropy values given should be seen as ranks of various models/solutions on an ordinal scale.

B. Correntropy based sampling procedure

In this section we propose a correntropy–based, hierarchical, binary density preserving splitting procedure. The correntropy given by Equation 12 is a function differentiable with respect to both x_i and y_i, which unfortunately is not

the case for the generalized function given by Equation 14. Moreover, none of them is differentiable with respect to the indices i and j, which are the only variables that can be manipulated within the splitting process. Gradient driven optimization procedure is thus not straightforward, hence we have reverted to a greedy, locally optimal approach.

Since correntropy is being estimated by a scaled sum of Gaussians, it reaches a maximum when all components of the sum reach their maximal values. In case of a single Gaussian function, the maximum is reached at 0, so the closer x_i and y_j are in Equation 14, the higher $V(X,Y)$ will be. This immediately suggests an iterative, binary splitting procedure of a dataset Z into datasets X and Y, which at each step selects two instances z_i and z_j so that:

$$i,j = \underset{i,j}{\operatorname{argmin}} \parallel z_i - z_j \parallel \tag{15}$$

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 $V(X,Y)$, that is the correntropy between the two new datasets. Due to the way correntropy is calculated for sets with various cardinalities however, it also indirectly maximizes $V(X,Z)$ and $V(Y,Z)$. As a result, newly obtained datasets are splits with distributions maximally similar to each other and to the distribution of the original dataset. To obtain more than 2 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:

$$d_{iX} + d_{jY} \geq d_{jX} + d_{iY} \Rightarrow X = X \cup z_i,\ Y = Y \cup z_j \tag{16}$$
$$d_{iX} + d_{jY} < d_{jX} + d_{iY} \Rightarrow X = X \cup z_j,\ Y = Y \cup z_i \tag{17}$$

For classification problems, the splitting procedure can be executed in either 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 turn and in separation from the rest. In other words the dataset is being split class by class. 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. Similar remark applies to estimation of correntropy, which can also be calculated in a class–wise (supervised) or class–less (unsupervised) mode.

The computational complexity of the DPS approach is of the order $O(N^2/2)$, as the most time consuming operation is calculation of pairwise distances between all N instances. This is however negligible when compared to the complexity of most training algorithms.

V. Experiments

The experiments have been conducted on 20 publicly available datasets using a total of 16 different classifiers. The datasets used come from the UCI Machine Learning Repository [3], the ELENA database [1] and the PRTools Pattern Recognition Toolbox for MATLAB [8]. The details of datasets are given in Table I. The star symbol in the '#obj/attr' column denotes the number of instances actually used in the experiments, sampled randomly from the whole, much bigger dataset in order to keep the experiments computationally tractable. The numbers given in brackets in the '#class' column denote the number of classes used, as we have removed the classes with less than 16 instances due to the problems it has caused during stratified sampling within the cross–validation scheme. The classifiers used come from the PRTools toolbox and their list is given in Table II.

The goal of the experiments was to compare the error estimation accuracy of cross–validation (CV) and density preserving sampling approaches (DPS) as well as to test the stability of both error estimators. We have followed a similar approach to that outlined in [12]. 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 has been calculated within a 10 times repeated 8–fold cross–validation 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 wrong the things can go with CV.

Three 8–fold DPS estimates are also given – DPS–S (using class label information), DPS–U (ignoring class label information) and DPS–SU (averaged over DPS–S and –U).

A. Toy problem

The analysis starts with Cone–torus, a synthetic dataset first used in [13]. The dataset has been chosen as it is two–dimensional, which allows for visualisation of the results and has been extensively used in our previous studies due to its well known properties. The Cone–torus dataset consists of 3 classes, with instances generated from 3 differently shaped distributions: a cone, half a torus, and a normal distribution. The analysis of the scatter plots of DPS and CV folds[2] have revealed, that for DPS the classes tend to preserve their shapes – the half torus for example was clearly visible in 7 out of 8 folds, while for CV only in 4 or 5. This was also

[2]The plots are not presented here due to space constraints but they are available at http://www.budka.co.uk.

TABLE I

DATASET DETAILS

abbr	name	source	#obj/attr	#class
can	Breast cancer Wisconsin	UCI	569/30	2 (2)
cba	Chromosome bands	PRTools	1000*/30	24 (24)
chr	Chromosome	PRTools	1143/8	24 (23)
clo	Clouds	ELENA	1000*/2	2 (2)
cnc	Concentric	ELENA	1000*/2	2 (2)
cnt	Cone–torus	[13]	800/3	2 (2)
dia	Pima Indians diabetes	UCI	768/8	2 (2)
ga2	Gaussians 2d	ELENA	1000*/2	2 (2)
ga4	Gaussians 4d	ELENA	1000*/4	2 (2)
ga8	Gaussians 8d	ELENA	1000*/8	2 (2)
ion	Ionosphere radar data	UCI	351/34	2 (2)
let	Letter images	UCI	1000*/16	26 (26)
liv	Liver disorder	UCI	345/6	2 (2)
pho	Phoneme speech	ELENA	1000*/5	2 (2)
sat	Satellite images	UCI	1000*/36	6 (6)
seg	Image segmentation	UCI	1000*/19	7 (7)
shu	Shuttle	UCI	1000*/9	7 (3)
syn	Synth–mat	[19]	1250/2	2 (2)
tex	Texture	ELENA	1000*/40	11 (11)
veh	Vehicle silhouettes	UCI	846/18	4 (4)

TABLE II

CLASSIFIER DETAILS

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
perlc	Linear Perceptron Classifier
rbnc	Radial Basis Function Neural Network Classifier

well reflected by the mean value of correntropy 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 1. The black solid line represents the boundaries of a classifier trained using DPS–S folds, while 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 seem very unstable and can change shape radically.

B. Benchmark datasets

1) Correntropy: Figure 2 presents the values of averaged correntropy between the original dataset and 8 folds generated using DPS and CV, for all 20 datasets used in the experiment. Note, that although the correntropy has been

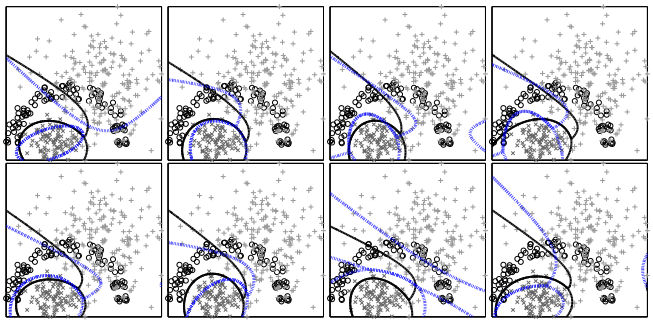

Fig. 1. Cone–torus – decision boundaries for qdc trained on DPS–S (solid line) and CV (dotted line) folds

normalized to the $0 \div 1$ range, according to our earlier argument the values represent an ordinal scale.

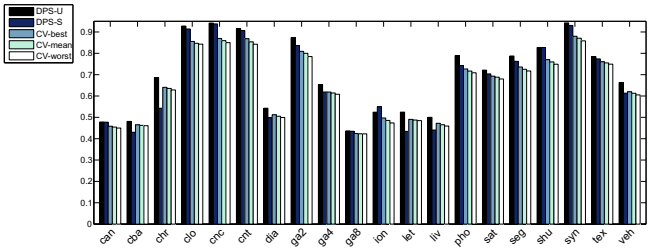

Fig. 2. Mean correntropy between each fold and the original dataset

The correntropy between the DPS folds and the original dataset is almost always higher than in the case of the CV folds, even for the best CV split out of 10. This is not surprising since the DPS splits have been obtained by maximization of correntropy. In some cases however, the supervised approach DPS–S falls behind CV. It happens in general when dealing with datasets with a large number of classes (e.g. cba – 24 or chr – 23), which must necessarily be small. This constraints the optimization process resulting in lower values of correntropy.

The picture is very similar for the between–fold correntropy depicted in Figure 3, where DPS–U is again an unquestionable leader, usually followed by DPS–S, except for the datasets with large numbers of classes.

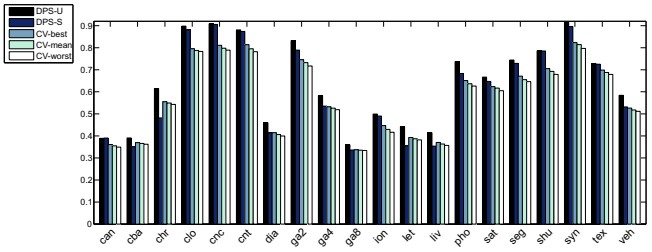

Fig. 3. Mean between–fold correntropy

2) Bias: The mean absolute bias for both DPS and CV can be seen in Figures 4 and 5. The DPS approach has a bias comparable to the mean CV result, with slight advantage of the latter for roughly half of the datasets. DPS estimates

are however never as biased as the worst–case CV scenario, yet the result was achieved with roughly 10 times less computations.

A summary of the results can be found in Table III. DPS–U has on average a lower bias than DPS–S by 0.0015 (mean) and 0.0005 (median), while DPS–SU falls in the middle, due to the way it was calculated. When compared to the CV, the best of the three DPS estimators is worse by just 0.0003 (both mean and median). Note, that the spread (the difference between maximal and minimal value) of the CV estimate is more than 1.3 times bigger than the best–case scenario bias.

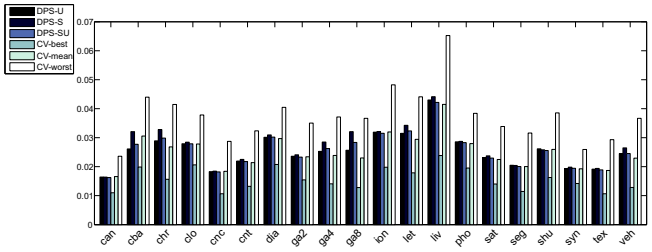

Fig. 4. Mean absolute bias (averaged over all classifiers)

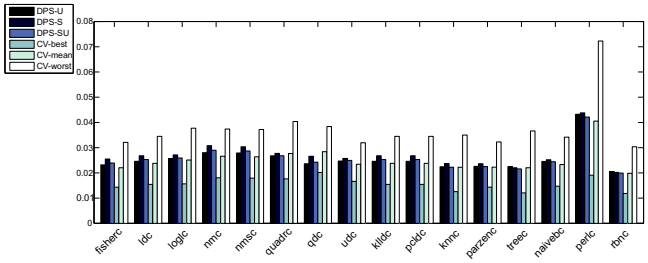

Fig. 5. Mean absolute bias (averaged over all datasets)

TABLE III
BIAS AND VARIANCE SUMMARY (FOR ALL DATASETS AND CLASSIFIERS)

	DPS U	**DPS S**	**DPS SU**	**CV best**	**CV mean**	**CV worst**
BIAS-mean	0.0256	0.0271	0.0263	0.0157	0.0251	0.0375
-median	0.0240	0.0245	0.0242	0.0148	0.0237	0.0342
VAR-mean	0.0384	0.0304	0.0344	0.0260	0.0429	0.0618
-median	0.0377	0.0281	0.0322	0.0264	0.0431	0.0615

3) Variance: The standard deviation of error estimates can be seen in Figure 6 (averaged over all classifiers) and Figure 7 (averaged over all datasets). Out of all three DPS approaches, now DPS–S takes the lead, with the variance comparable even to the best–case CV scenario for some datasets. All three DPS approaches also have the variance lower than the averaged value of 10 CV runs. According to the summary given in Table III, for DPS–S and CV–mean the difference reaches 0.0125 (mean) and 0.0150 (median).

Note, that in terms of bias, DPS–U was the leading approach out of the three, while now DPS–S performs better. We thus have two low complexity error estimates, one with low bias but higher variance, the second with higher bias but low variance. In practice, the third option – DPS–SU might be worth considering, as it brings the best of both worlds – a bit higher bias than CV (0.0012 – mean / 0.0005 – median) and much lower variance (0.0085 / 0.0109), at about 20% of the computations required by 10 times repeated CV.

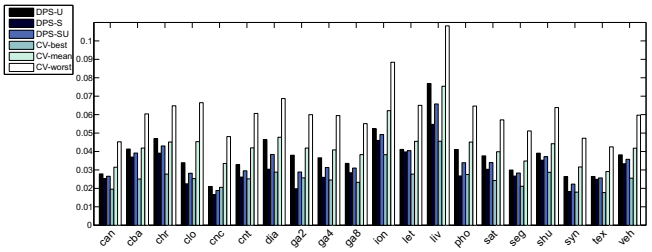

Fig. 6. Standard deviation of error estimate (averaged over all classifiers)

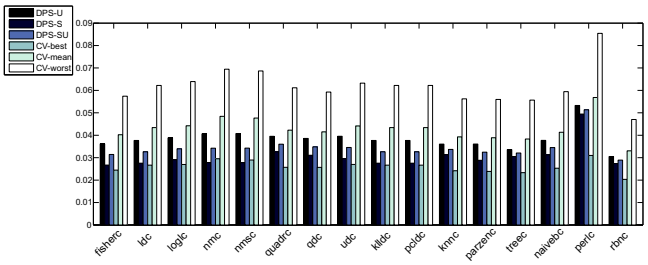

Fig. 7. Standard deviation of error estimate (averaged over all datasets)

VI. DISCUSSION

The presented Density Preserving Sampling procedure is a very attractive alternative for the commonly used cross–validation technique for a number of reasons.

For the purpose of the generalization error estimation, k–fold cross–validation 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 worst–case scenario estimate, which as demonstrated in this paper can be highly biased and can have a large variance. The need of running the procedure multiple times makes it computationally expensive, forcing the researchers to seek compromise elsewhere, for example by not calculating the full gradient during optimization or taking other shortcuts, which negatively influence optimality of the solutions found. The DPS procedure proposed in this paper is however deterministic. It thus does not need to be repeated in order to improve the quality of the error estimate, at the same time producing results comparable to repeated cross–validation when it comes to bias, and superior to CV in terms of the variance of obtained estimates. Yet it all happens at 5–10 times lower computational cost.

Another related application area of CV is parameter estimation. Since for some models the objective function is not always differentiable with regard to all its parameters,

the optimization procedure must resort to a search in the parameter space. One example of such situation might be the k–NN classifier, for which the number of nearest neighbours k is usually being set by testing a number of possible values using cross–validation. In such case, as the search itself might be very costly depending on the dimensionality of the search space, 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. 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 technique to cross–train multiple models and select the best performing one. The cross–training procedure is analogous to cross–validation error estimation, that is the dataset is divided randomly into k folds, which are then put aside one by one to be used for verification of the model trained using the remaining $k-1$ folds. The difference is 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 estimation procedure (see plots of the decision boundaries in Figure 1) with an unstable learning method, which in an unfavorable case 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 application of random sampling procedures we want to discuss here 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, which is a sign of overfitting. The risk of using unrepresentative validation set is obvious in this case. Although the behavior of DPS in conjunction with early stopping has not been addressed in this paper, it forms an interesting and promising future research direction, which will be investigated in our further work.

VII. Conclusions

The correntropy–based density–preserving data sampling (DPS) procedure developed and investigated in this paper is an interesting alternative for widely used cross–validation technique in many applications. Unlike CV, DPS is a deterministic method, 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 the previous section. The experiments conducted using a set of publicly available benchmark datasets and standard classifiers have revealed that:

- For generalization error estimation, DPS is slightly more biased than 10 times repeated cross–validation but has much lower variance, comparable with 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 using DPS folds are much more stable than in the case of a single cross–validation folds, which is the result of representativeness of the subsets generated by DPS.

Further research will focus on application of DPS to early stopping and model selection, as well as on using correntropy as a measure of diversity of datasets for building ensemble models. The results will be published in an extended version of this paper, which will also include experimental results not presented here due to limited presentation space.

References

[1] ELENA database, 1995.

[2] A. Antos, L. Devroye, and L. Gyorfi. Lower bounds for Bayes error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7):643–645, 1999.

[3] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[4] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1995.

[5] T.M. Cover. Learning in Pattern Recognition., 1968.

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

[7] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification 2nd ed.* Wiley–Interscience, 2000.

[8] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D.M.J. Tax, and S. Verzakov. PR-Tools 4.1, a MATLAB toolbox for Pattern Recognition, 2007.

[9] B. Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[10] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, pages 4–37, 2000.

[11] K.H. Jeong and J.C. Principe. Enhancing the correntropy MACE filter with random projections. *Neurocomputing*, 72(1-3):102–111, 2008.

[12] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. Citeseer, 1995.

[13] L.I. Kuncheva. *Fuzzy classifier design*. Physica Verlag, 2000.

[14] W. Liu, P.P. Pokharel, and J.C. Principe. Correntropy: A Localized Similarity Measure. *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 4919–4924, 2006.

[15] W. Liu, PP Pokharel, and JC Principe. Error Entropy, Correntropy and M-Estimation. In *Machine Learning for Signal Processing, 2006. Proceedings of the 2006 16th IEEE Signal Processing Society Workshop on*, pages 179–184, 2006.

[16] W. Liu, P.P. Pokharel, and J.C. Principe. Correntropy: properties and applications in non-Gaussian signal processing. *IEEE Transactions on Signal Processing*, 55(11):5286, 2007.

[17] J.C. Principe, D. Xu, and J. Fisher. Information theoretic learning. *Unsupervised Adaptive Filtering*, pages 265–319, 2000.

[18] J.C. Principe, D. Xu, Q. Zhao, and J.W. Fisher. Learning from Examples with Information Theoretic Criteria. *The Journal of VLSI Signal Processing*, 26(1):61–77, 2000.

[19] BD Ripley. Pattern recognition and neural networks. 1996.

[20] I. Santamaría, P. Pokharel, and J.C. Principe. Generalized correlation function: Definition, properties, and application to blind equalization. *IEEE Transactions on Signal Processing*, 54(6):2187, 2006.

[21] S.M. Weiss and C.A. Kulikowski. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1991.