Machine Learning with Known Input Data Uncertainty Measure

Wojciech M. Czarnecki, Igor T. Podolak

Machine Learning with Known Input Data Uncertainty Measure

Wojciech M. Czarnecki[1] and Igor T. Podolak[2]

[1] Faculty of Mathematics and Computer Science,
Adam Mickiewicz University in Poznan,
w.czarnecki@amu.edu.pl
[2] Faculty of Mathematics and Computer Science,
Jagiellonian University
igor.podolak@uj.edu.pl

Abstract. Uncertainty of the input data is a common issue in machine learning. In this paper we show how one can incorporate knowledge on uncertainty measure regarding particular points in the training set. This may boost up models accuracy as well as reduce overfitting. We show an approach based on the classical training with jitter for Artificial Neural Networks (ANNs). We prove that our method, which can be applied to a wide class of models, is approximately equivalent to generalised Tikhonov regularisation learning. We also compare our results with some alternative methods. In the end we discuss further prospects and applications.

Keywords: machine learning, neural networks, classification, clustering, jitter, uncertainty, random variables

1 Introduction

Uncertainty is a popular phenomenon in several life areas, such as medicine, image processing, linguistics, robotics, etc. Uncertainty types relate to modelled system input, output, and their representation. Previously much attention was paid mainly to the expected output, e.g. by taking into account class label together with its probability [1]. The output label uncertainty has to be approached when using an active learning methodology, e.g. with multiple oracles [2].

In the context of input data uncertainty, most frequently addressed issue is the missing attributes problem. Solutions include, e.g. explicitly modelling the expected risk [3], data imputation using EM [4], use of the grey information theory [5], etc.

On the other hand, the problem of input data uncertainty (meaning that input vectors are enriched with some kind of certainty distribution) is addressed less frequently. This is worth considering in such areas as robotics, biology, medicine, pharmacology, etc. For example, in robotics we may imagine a situation where a robot detecting an object using some methods of measurements, can also estimate its certainty of measurement correctness. It might depend on several factors: sensor type, localisation, time spent on taking the measurement, environment conditions, etc. Analogously, in experimental science, observation induced error can be estimated [6].

In this paper we investigate how one can exploit the measurement uncertainty if it is given directly as a probability density function. First, we shall propose a method of incorporating this kind of knowledge into iterative learning algorithms. Then we shall show some theoretical properties. Some experiments shall be performed as a proof of concept. Conclusions follow.

2 Methods

There are different solutions possible for the problem of learning tasks when the input attribute measurements are inaccurate. The most obvious would be to generate incorrect data. This might prove to be troublesome and the learning process might be intractable.

It is different when the inaccuracies are known in advance and given in the form of probability of distributions of points being sampled in the neighbourhood of the exact attribute value. The distribution might be given in the form of, e.g. a normal distribution with zero mean and a given covariance matrix (see Fig. 1). Given such additional knowledge, and a number generator, we may sample points to be used during learning. As we will show, when using this approach, data points from the whole input space may be labelled, and learning belongs to a class of generalised Tikhonov's regularisation [7].

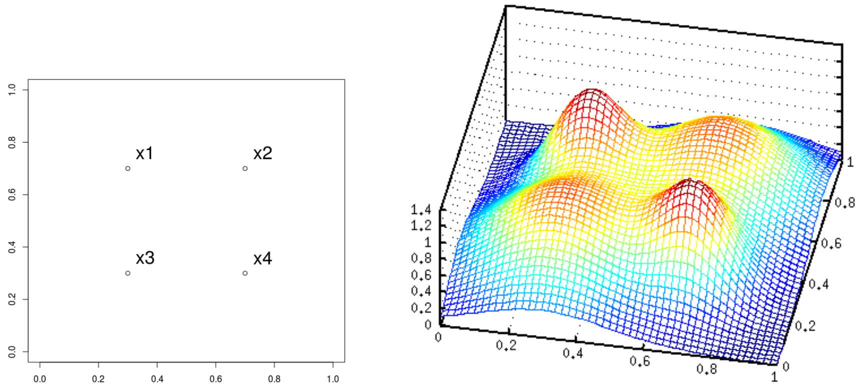

Fig. 1. Sample training set with four data points (on the left) transformed by adding to the $x1$ and $x4$ points normal distribution with zero mean and variance 0.04 and the same distribution with variance 0.01 to the remaining two points (on the right). Labels are omitted for clarity.

This method is, naturally, applicable only in the case of iterative methods, like a steepest descent with a defined energy function. Therefore, we shall use, as an example, feed–forward neural networks. It has to be clearly pointed, that other method, e.g. clustering, may benefit from this model.

2.1 Sampling the input space

To provide data needed for learning, we need to sample the input space. Each training example is, actually, a triple (x_i, t_i, f_i) where x_i is a P-dimensional numeric attribute vector, t_i is its target label (either for a regression or classification task), while f_i is a P-dimensional probability density function. This pdf might be, e.g. a normal distribution $\mathcal{N}(0, Q_i)$, where Q_i is the covariance matrix, and gives the likelihood the actual attribute measurement is biased from its true value x_i. The actual data points to be classified by a trained model shall be $x_i + \nu$, where ν is the noise vector. In the following sections we assume that all the distributions are zero-mean and normalised.

We have two basic methods of sampling data, described in Table 1, which may easily be shown to be equivalent.

Table 1. Data sampling methods

draw from f_i individually	draw from sum of f_i
1. draw an index i corresponding to example x_i with label t_i 2. draw a random variable ν with some given probability f_i 3. return a training example $(x_i + \nu, t_i, f_i)$	1. draw a data point $\widetilde{x}$ with probability given with $\sum_i f_i$ 2. draw a label $t = t_i \sim f_i(\widetilde{x}) / \sum_j f_j(\widetilde{x})$ 3. $\nu = \widetilde{x} - x_i$ 4. return a training example $(x_i + \nu, t_i, f_i)$

2.2 Training with jitter

Training with jitter is a method in which data is corrupted with noise [9]. This approach is designed to help generalisation. When training, the examples are taken from the training set with some small noise term ν added. Let $\widetilde{x} = x_i + \nu$. The target label is taken to be that of x_i, i.e. $t(\widetilde{x}) = t_i$. Thus, the expected value of the target is $E[t(\widetilde{x} - \nu)|\widetilde{x}] = \sum_i t_i P(i|\widetilde{x})$, where $P(i|\widetilde{x})$ denotes the probability that $\widetilde{x}$ is a noisy exemplification of example x_i. In other words, the labels are actually defined for all the points in the attribute space.

As Reed et al. [9] show, the training is equivalent to minimising the loss function $\ell(\widetilde{x}) = \left[\sum_i t_i P(i|\widetilde{x}) - y(\widetilde{x})\right]^2$ (provided a least-squares approach is used). This shows, that this training supports generalisation, which has been proven in several experiments [10, 11].

Our method is to extend the training with jitter approach to the case with many different, but known, noise distributions. Such approach needs a theoretical justification, therefore we shall use the Tikhonov regularisation model as a base, which is a well studied and grounded method in machine learning. This implies many well known properties and advantages of this approach. Intuitively, thanks to distribution implied labeling of all input space data points, our method adds regularisation.

2.3 Input uncertainty training as a regularisation method

Proposition 1 *Let $\mathcal{M}$ be a least squares minimisation procedure. If we use a training set with added zero-mean, normalised random noise coming from known distributions defined separately for each training data point, then there exists a procedure $\mathcal{M}'$ approximately equal to $\mathcal{M}$, such that $\mathcal{M}'$ belongs to a class of generalised Tikhonov's regularisation.*

Proof. Let the training set be $\{(x_i, t_i, f_i | i = 1, \dots, N)\}$, where f_i pdfs are parameterised with Q_i covariance matrices. When using a gradient descent teaching function to train an ANN with weight matrix W, we may define the cost function as

$$E(W) = \sum_{i=1}^{N} \int\limits_{\mathbb{R}^K} \big[t(x_i) - y(x_i + \nu)\big]^2 P(x_i) f_i(x_i + \nu) d\nu, \tag{1}$$

where ν is the noise vector, and $P(x_i) = \int\limits_{\mathbb{R}^K} P(i|x_i + \nu)$. We can easily expand it

$$\begin{aligned} E(W) &= \textstyle\sum_{i=1}^{N} \int\limits_{\mathbb{R}^K} \left[t^2(x_i) - 2t^2(x_i)y^2(x_i+\nu) + y^2(x_i+\nu)\right] P(x_i) f_i(x_i+\nu) d\nu \\ &= \textstyle\sum_{i=1}^{N} \big[t^2(x_i) \int\limits_{\mathbb{R}^K} f_i(x_i+\nu)d\nu - 2 \int\limits_{\mathbb{R}^K} t^2(x_i)y^2(x_i+\nu)f_i(x_i+\nu)d\nu \\ &\qquad + \int\limits_{\mathbb{R}^K} y^2(x_i+\nu)f_i(x_i+\nu)d\nu\big] P(x_i) \\ &= \textstyle\sum_{i=1}^{N} \big[t^2(x_i) \int\limits_{\mathbb{R}^K} f_i(x_i+\nu)d\nu - 2t(x_i) \int\limits_{\mathbb{R}^K} y(x_i+\nu)f_i(x_i+\nu)d\nu \\ &\qquad + \int\limits_{\mathbb{R}^K} y^2(x_i+\nu)f_i(x_i+\nu)d\nu\big] P(x_i) \end{aligned} \tag{2}$$

If the ν noise is small, we can approximate using the Taylor expansion

$$y(x+\nu) = y(x) + \left(\frac{\partial y}{\partial x}\right)^T \nu + O(\nu^2)$$

with derivatives computed at $\nu = 0$. For sufficiently small ν we can omit the last term. Substituting into (2) and expanding

$$
\begin{aligned}
E(W) \simeq \textstyle\sum_{i=1}^{N} \big[t^2(x_i) & \int\limits_{\mathbb{R}^K} f_i(x_i+\nu)d\nu - 2t(x_i) \int\limits_{\mathbb{R}^K} y(x_i+\nu) f_i(x_i+\nu)d\nu \\
& -2t(x_i) \int\limits_{\mathbb{R}^K} \left(\frac{\partial y}{\partial x}\right)^T f_i(x_i+\nu)d\nu + \int\limits_{\mathbb{R}^K} y^2(x_i+\nu) f_i(x_i+\nu)d\nu \\
& +2 \int\limits_{\mathbb{R}^K} y(x_i+\nu) \left(\frac{\partial y}{\partial x}\right)^T \nu f_i(x_i+\nu)d\nu \\
& + \int\limits_{\mathbb{R}^K} \left(\frac{\partial y}{\partial x}\right)^T \nu\nu^T \left(\frac{\partial y}{\partial x}\right) f_i(x_i+\nu)d\nu \big] P(x_i)
\end{aligned}
$$

Since $\int\limits_{\mathbb{R}^K} f_i(x_i+\nu)d\nu = 1$

$$
\begin{aligned}
E(W) &\simeq \sum_{i=1}^{N} \big[t^2(x_i) - 2t(x_i)y(x_i+\nu) + y^2(x_i+\nu) \big] P(x_i) \\
&\quad + \sum_{i=1}^{N} [\left(\frac{\partial y}{\partial x}\right)^T \nu\nu^T \left(\frac{\partial y}{\partial x}\right)] P(x_i) \\
&= \sum_{i=1}^{N} \big[t(x_i) - y(x_i+\nu) \big]^2 P(x_i) + \sum_{i=1}^{N} [\left(\frac{\partial y}{\partial x}\right)^T Q_i \left(\frac{\partial y}{\partial x}\right)] P(x_i) \\
&= \sum_{i=1}^{N} \big[t(x_i) - y(x_i+\nu) \big]^2 P(x_i) + \sum_{i=1}^{N} \left\| \frac{\partial y}{\partial x} \right\|_{Q_i}^2 P(x_i), \qquad (3)
\end{aligned}
$$

where Q_i is the covariance matrix corresponding to the f_i pdf, and $\|x\|_{Q_i}^2$ is a weighted norm $x^T Q_i x$. Equation (3) is a Tikhonov regularisation. □

Observation 1 *There are four different possibilities in case of $\mathcal{N}(0, Q_i)$:*

1. *all pdfs are identical and non-correlated $Q_i = \sigma^2 I$ in that case the regularising term reduces to $\sigma^2 \sum_{i=1}^{N} \|\frac{\partial y}{\partial x}\|^2 P(x_i)$ which corresponds to results in [8], the variance determines the relevance of the regularising term,*
2. *all pdfs are identical, but correlated $Q_i = Q$; the regularising term reduces to $\sum_{i=1}^{N} \|\frac{\partial y}{\partial x}\|_Q^2 P(x_i)$, similarly, the variances determine the regularising factor but with different strength depending on the dimension,*
3. *pdfs are different and uncorrelated $Q_i = \sigma_i^2 I$; the regularising term reduces to $\sum_{i=1}^{N} \sigma_i^2 \|\frac{\partial y}{\partial x}\|^2 P(x_i)$, variance determines the relevance of the regularisation, but its strength depends on the distance from training points: in neighbourhood of training data with higher variance, this term is more important; in areas of lower certainty the function smoothness becomes more important than its exact fitting,*

4. pdfs are correlated and different, the regularising term has the basic form $\sum_{i=1}^{N} \| \frac{\partial y}{\partial x} \|_{Q_i}^2 P(x_i)$*, the strength depends now both on the distance from training points and the dimensions.*

3 Experiments

During our experiments we have used the Encog Machine Learning Framework [12] for which we developed classes to treat the input data as random variables of known distributions. Sampling is derived using the Apache Commons Mathematics Library [13] to ensure high quality of this step. We focus on the three layered feed forward neural network with sigmoidal activation functions as our model, trained using classic backpropagation algorithm with momentum, as it is a simple, well known model that can be trained in the iterative fashion. Training set was resampled from the f_i distributions after each iteration (see Fig. 2 for details). In fact, we do not need the exact closed form equations of f_i but rather we just require a sampling method (some blackboxes s_i that provides us with samples $x_i + f$, with f coming from f_i).

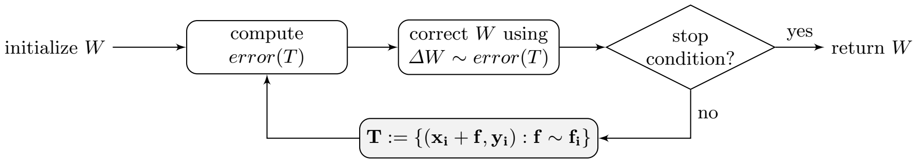
initialize W
compute $error(T)$
correct W using $\Delta W \sim error(T)$
stop condition?
yes
return W
no
$\mathbf{T} := \{(\mathbf{x_i} + \mathbf{f}, \mathbf{y_i}) : \mathbf{f} \sim \mathbf{f_i}\}$

Fig. 2. Flowchart of the proposed method. Classical iterative ANNs learning techniques include repeated weights (W) corrections according to the current *error* computed on the training set T (white blocks). In our model, the only required modification is to resample the T set from respective distributions f_i after each iteration (gray block).

Three toy experiments are described below, namely (σ_{ik}^2 is the variance of the k-th example in the i-th class):

1. Training set consisting of 4 points, two of one class (triangles) located in $(0.3, 0.3)$ and $(0.7, 0.7)$ and two of the second class (circles) located in $(0.3, 0.7)$ and $(0.7, 0.3)$. Assumed distributions are zero mean Gaussian distributions with variance $\sigma_1^2 = 0.04$ and $\sigma_2^2 = 0.01$ respectively.
2. Training set consisting of 2 points, one class (triangles) located in $(0.3, 0.3)$ and second class (circles) located in $(0.7, 0.7)$. Assumed distributions are zero mean Gaussian distributions with variance $\sigma_1^2 = 0.09$ and $\sigma_2^2 = 0.01$ respectively.
3. Training set consisting of 10 points, five of one class (triangles) located in $(0.3{+}0.1k, 0.3)$ and five of the second class (circles) located in $(0.3{+}0.1k, 0.7)$

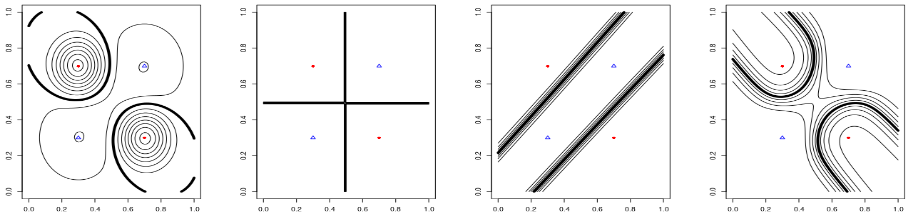

Fig. 3. Results of experiment 1, from left: Bayesian classification, Voronoi diagram, neural network classification using simple learning (discarding distribution information), our method. Thick lines indicate decision boundaries.

for $k \in \{0, 1, 2, 3, 4\}$. Assumed distributions are zero mean Gaussian distributions with variances $\sigma_{1k}^2 = 0.025k^2$ and $\sigma_{2k}^2 = 0.036k^2$ respectively.

Fig. 3 shows results of our first experiment for points classification from $[0, 1]^2$. Additional information regarding input data point uncertainty helps this model in defining more adequate decision boundary. Four points used in this example are fully symmetric, therefore decision boundaries found by simple neural network are parallel lines between elements of different classes. Different variances in each of the class forced neural network to bend its decision boundary towards points with higher certainty (as it is more probable that points in the unknown parts of space are parts of the second distribution).

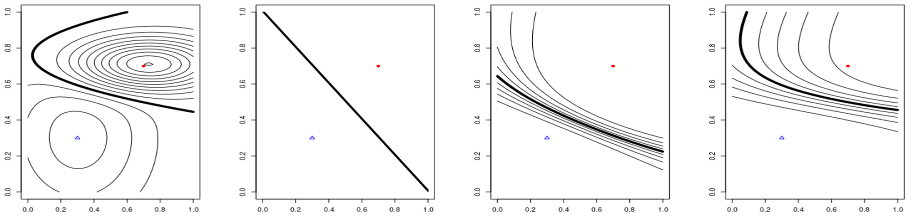

Fig. 4. Results of experiment 2, from left: Bayesian classification, Voronoi diagram, neural network classification using simple learning (discarding distribution information), our method. Thick lines indicate decision boundaries.

Fig. 4 captures similar observations like the previous one, but as the Voronoi diagram for this dataset is also a result of the support vector machine (SVM) classifier trained on such data — it shows how this additional information about relative difference in our certainty about particular data points affect the decision boundary. In this particular example model obtained by training neural network is a better approximation of the underlying distribution (generalisation) than ones returned by both SVM and an ANN.

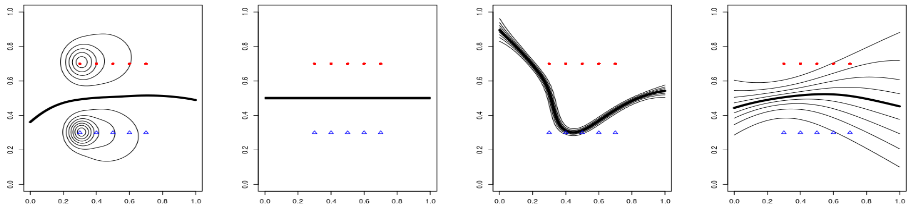

Fig. 5. Results of experiment 3, from left: Bayesian classification, Voronoi diagram, neural network classification using simple learning (discarding distribution information), our method. Thick lines indicate decision boundaries.

With more data points available in the training set the effect of uncertainty measure is even more noticeable. Fig. 5 shows result of our third experiment, where one can observe the overfitting problem of the classical neural network while sampling method achieves nice, smooth decision boundary approximating underlying distribution. In particular, Fig. 6 shows results of four different runs of experiment 3, where one can see strength of the regularisation induced by uncertainty measure in comparison to the overfitting and complete indeterministic behaviour of simple neural network learning.

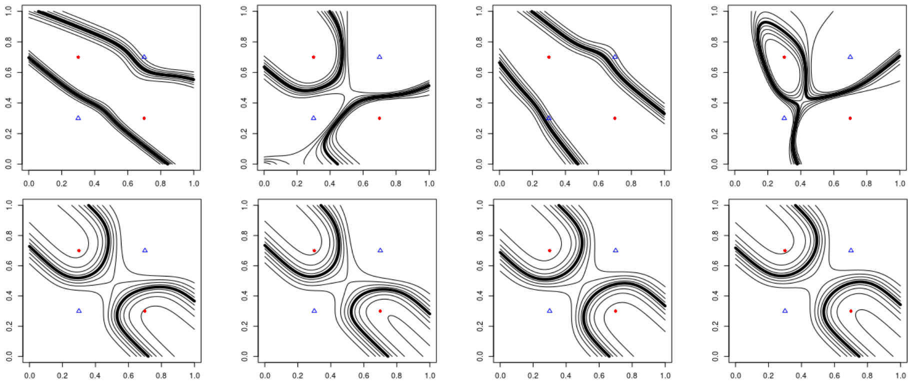

Fig. 6. Example of overfitting of a simple ANN in experiment 1 (top row) and almost indistinguishable results of multiple runs of our sampling method (bottom row)

Fig. 7 shows mean squared error during one of the experiments 1 and 3 runs using random sampling method as compared to simple training. As one can see - learning curve of the experiments 1 dataset behaves quite similarly - after short amount of time it rapidly decreases and stays in the local minimum. Although the main difference is the curve smoothness - rapid changes in its value make it unreasonable to use simple thresholding as a stop condition for our method.

There are at least three possible ways to overcome this limitation:

- train as long as possible, as Proposition 1 and experiments clearly show that such training is strongly regularised so overfitting is highly unlikely,
- use some statistics of errors instead of its value (even moving averages behaves well in our experiments),
- if we have knowledge about analytic form of our distributions sufficient to compute the regularisation term from Eq. (3) then (according to Proposition 1) we can train as long as regularised form of error equation would be higher than some threshold.

It is worth noting that such an approach actually speeds up the convergence as greedy optimisation of the smoothed function is much more rapid (see right part of Fig. 7 for reference).

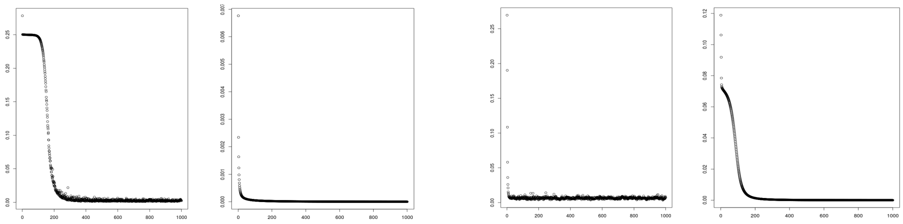

Fig. 7. Comparison of the error curves during training, from the left: experiment 1-our method, experiment 1-simple training, experiment 3-our method, experiment 3-simple training. On x axis - training step, on y axis - mean squared error on training data

4 Conclusions and future work

In this paper we have proposed methods of incorporating knowledge on input data uncertainty into the learning process for a range of machine learning models, ANNs in particular. We have proved, both theoretically and empirically, that this approach is valid and yields valuable results. In particular, expressing it as a generalised Tikhonov regularisation shows its immunity to the overfitting problem.

It remains an open question what are the exact conditions which the data pdfs have to fulfil in order to insure convergence to the minimum of the regularising functional. In particular, it is obvious that uniform distribution, e.g. with $n = 2$, may lead to oscilations. Therefore, can we use discrete distributions in general?

The future work will concentrate on application of this methodology to clustering problems, e.g. Kohonen or K-means algorithms, with particular interest in convergence requirements. We also want to extend the active learning paradigm in direction of widening the spectrum of queries [14]. In addition to just asking for a label of a particular data point, we could also query for a corrected measurement, which should have higher certainty and smaller variance. Both have applications in, e.g. robotics.

We also plan to investigate applicability of such model in complex classifiers like Hierarchical Classifier with Overlapping Classes (HCOC [15]) which could use it as a node level supervised classifier, with decresing noise variance on deeper tree levels.

5 Acknowledgment

This work is partially financed by the National Science Centre grant *Hierarchical Classifier model. Fusion of supervised learning and non-supervised clustering into sub-problems. Theoretical problems and practical examples* (N N519 654480).

References

1. Niaf, E., Flamary, R., Lartizien, C., Canu, S.: Handling uncertainties in SVM classification. In: IEEE Statistical Signal Processing Workshop, pp. 757–760 (2011)
2. Ni, E.A., Ling, C.X.: Active learning with c-certainty, Advances in Knowledge Discovery and Data Mining. LNCS, vol. 7301, pp. 231-242. Springer, Heidelberg (2012)
3. Pelckmans, K., De Brabanter, J., Suykens, J.A.K., De Moor, B.: Handling missing values in support vector machine classifiers. Neural Networks 18, 684–692 (2005)
4. Zhang, S.S., Wu, X., Zhu, M.: Efficient missing data imputation for supervised learning. In: 9th IEEE Int. Conf. on Cognitive Informatics, pp. 672–679 (2010)
5. Han, B., Xiao, S., Liu, L., Wu, Z.: New methods for filling missing values by grey relational analysis. Int. Conf. on Artificial Intelligence, Management Science and Electronic Commerce, pp. 2721-2724 (2011)
6. Coleman, H.W., Steele, W.G.: Experimentation, validation and uncertainty analysis for engineers. John Wiley and Sons, (2009)
7. Tikhonov, A.N., Arsenin, V.Y.: Solutions of ill–posed problems. V.H. Winston (1977)
8. Bishop, C.M.: Training with noise is equivalent to Tikhonov regularization. Neural Computation 7(1), 108–116 (1995)
9. Reed, R., Oh, S., Marks, R.J.: Regularization using jittered training data. In: IEEE Int. Joint Conf. on Neural Networks, pp. 147–152, IEEE Press (1992)
10. Sietsma, J., Dow, R.J.F.: Creating artificial neural networks that generalise. Neural Networks 4(1), 1481–1497 (1990)
11. Weigand, A.S., Rumelhart, D.E., Huberman, B.A.: Generalization by weight elimination applied to currency exchange rate prediction. In: Int. Joint Conf. on Neural Networks, pp. I–837 (1991)
12. Heaton, J. Programming neural networks with Encog 3 in Java. Heaton Research Inc. (2011)
13. Apache Commons Mathematics Library. `http://commons.apache.org/math`
14. Settles, B.: Active Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2012)

15. Podolak, I., Roman, A.: Theoretical Foundations and Experimental Results for a Hierarchical Classifier with Overlapping Clusters, Computational Intelligence 29(2), pp. 357–388 (2013)