Abstracts of the FLLL/SCCH Master and PhD Seminar

Top 2/8, Software Park Hagenberg
June 23, 2006

Software Competence Center Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 800
Fax +43 7236 3343 888
www.scch.at

Fuzzy Logic Laboratorium Linz-Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
Tel. +43 7236 3343 431
Fax +43 7236 3343 434
www.flll.jku.at

Program

Session 1 (Chair: Susanne Saminger) 9:00–10:00

9:00	Bettina Heise: *(Blind) Deconvolution revisited*
9:30	Werner Groißböck: *HDFormGen: A Fast Nonlinear Approximation Formula Generator for Very High Dimensional Data Based on Variable Selection and Genetic Programming*

Coffee Break 10:00–10:15

Session 2 (Chair: Holger Schöner) 10:15–11:15

10:15	Susanne Saminger: *A Note on Ordinal Sums of T-Norms and T-Subnorms on Bounded Lattices*
10:45	Thomas Natschläger: *The Machine Learning Framework for Mathematica: Creating interpretable, computational models from data*
11:15	*Closing of the seminar*

Blind Deconvolution

Abstract:
Deconvolution as an inverse problem is included in many signal and image processing tasks, as e.g. in seismic and ultrasound reflectivity measurements, in signal equalization and image processing. Closely related to this task are Blind Source Separation problems. In the latter problem we have at least as much signals as sources, whereas in deconvolution problems only a single time or space delayed signal is available. Dependent on the fact whether the PSF is known or unknown, we distinguish between blind and non-blind deconvolution.
The aim of deconvolution is the reconstruction (estimation û(k)) of the original signal *u(k)* disturbed by the system H(z) and degraded by noise *η(k)* from the measured signal *v(k)*.

Optimization algorithms can achieve the solution of this task. The quantity which may be optimized can be chosen in different way according to the given measurement problem. Least square approaches occasionally offer suboptimal results and may produce spurious unphysical solutions.

$$J_{MSE} = E\left\{e^2\right\} = E\left\{\left[u(m,n) - \hat{u}(m,n)\right]^2\right\} \rightarrow \min$$
$$with \quad \hat{u}(m,n) = \sum_k \sum_l \tilde{w}(m,n,k,l)\, v(k,l)$$

This can be improved by additionally constraints as positivity or band limitation. A further approach can be weighted least squares or preconditioning for improving stability.

$$\mathbf{Hu} = \mathbf{v} \Leftrightarrow \mathbf{H}^{\mathbf{T}}\mathbf{H}\,\mathbf{u} = \mathbf{H}^{\mathbf{T}}\mathbf{v} \Leftrightarrow \widehat{\mathbf{H}}\mathbf{u} = \widehat{\mathbf{v}}$$

$$Image\ mxn \Rightarrow \mathbf{H} : \dim^{m\circ n} x^{\ m\circ n}, \mathbf{u} : \dim^{m\circ n} x^1, \mathbf{v} : \dim^{m\circ n} x^1$$

$$Preconditioner: \quad \mathbf{B} = \frac{1}{\varepsilon}\mathbf{I}$$
$$Iteration: \mathbf{u^{k+1}} = \mathbf{u^k} - \mathbf{B}^{-1}(\widehat{\mathbf{H}}\mathbf{u}^k - \widehat{\mathbf{v}})$$

Discribing the imaging process by tensor **T**,

$$\mathbf{zTu} = \mathbf{v}\ , with\ \mathbf{zT} = \mathbf{H}, \mathbf{Tu} = \mathbf{E}^T$$

we can also solve the iteration for the (unknown) PSF **z** in the same way.

$$\mathbf{zE^T} = \mathbf{v} \Leftrightarrow \mathbf{zE^TE} = \mathbf{vE} \Leftrightarrow \mathbf{z}\breve{\mathbf{E}} = \breve{\mathbf{v}}$$
$$\mathbf{E(\hat{u})} : \dim^{\ m\circ n} x^{\ p\circ p}$$
$$\mathbf{v} : \dim^1 x^{m\circ n}, \mathbf{z} : \dim^1 x^{p\circ p}$$
$$\breve{\mathbf{E}} : \dim^{p\circ p} x^{\ p\circ p}$$
$$\breve{\mathbf{v}} : \dim^1 x^{p\circ p}$$

Using additionally statistical information f_V about the imaging process leads to maximum or minimum entropy H,

$$H(v) = -E\left\{\ln(f_V(v))\right\} = -\int_{-\infty}^{\infty} f_V(v)\ln(f_V(v))dv \rightarrow \max/\min!$$

maximum information I,

$$I(u,v) = H(v) - H(v \setminus u) = H(v) - H(\eta) \rightarrow \max!$$

or maximum likelihood L -deconvolution methods.

$$f_V(v,\theta), \quad \textit{estimate } \hat{f}_V(v,\theta)$$

$$J_{ML}(\theta) = E\left\{\ln \hat{f}_V(v,\theta)\right\} = \int f_V(v)\cdot \ln \hat{f}_V(v,\theta)\, dv \rightarrow \max!$$

$$\Leftrightarrow K(f_V \mid \hat{f}_V) = \int \ln\left(\frac{f_V(v)}{\hat{f}_V(v,\theta)}\right) f_V(v)dv \rightarrow \min!$$

In fluorescence imaging Poisson or Gaussian noise statistics given by the probability p is mostly assumed, depending on the number of photon counts.

$$p_P(v \mid u) = \prod_i \frac{\mu_i \exp(-\mu_i)}{N_i!} \quad \textit{with } \mu_i = \beta[\mathbf{Hu}]_i,\ N_i = \beta\, v_i$$

$$p_G(v \mid u) = c\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{Hu}-\mathbf{v}\|^2\right)$$

For stability, we often have to choose a proper regularization, which may be given by prior knowledge. For instance, we can assume smoothness or entropy reduction also as additional constraints.

$$p(u \mid v) = \frac{p(v \mid u)p(u)}{p(v)}$$

$$p_G(u) = c\exp\left(-\frac{1}{2\tau^2}\|\mathbf{C}(\mathbf{u}-\mathbf{m})\|^2\right); \quad \mathbf{C},\mathbf{m}\ \textit{regular.matrix resp.model}$$

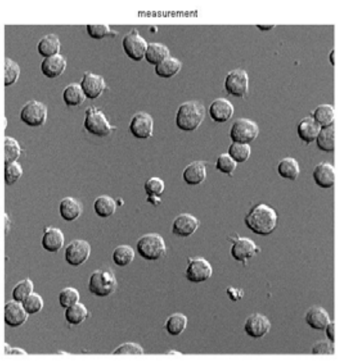
$$p_{En}(u) = c\exp(\rho H(|\mathbf{Cu}|,|\mathbf{Cm}|)) \quad \textit{with} \quad H(x,y) = \sum_i \left(x_i - y_i - x_i \ln\frac{x_i}{y_i}\right)$$

Simplified assuming Gaussian noise with $\mathcal{N}(0, \sigma_n)$ for the DIC images, we can optimize the likelihood $p(v|u)$ of the measured image v under condition of the theoretical image model *u*. Separating the problem in an optimization according to image noise statistics *M* and PSF model *N*, we can optimize the parameter of our PSF model given by two opposite Gaussians, which should simulate the differentiation property of DIC, with the values of height A_1, A_2 and spread σ_1, σ_2 displaced by shear τ.

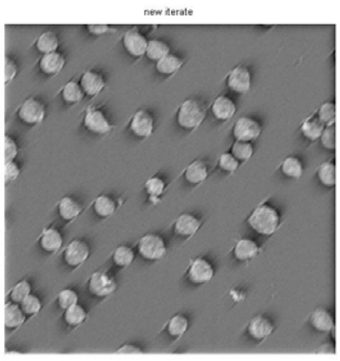
$$
\begin{aligned}
&L(u,\tau,A_1,A_2,\sigma_1,\sigma_2,\sigma_n \mid v) \overset{Separation}{\rightarrow} M,N \\
&M(u \mid v) \rightarrow \max! \\
&N(\tau,A_1,A_2,\sigma_1,\sigma_2,\sigma_n \mid v) \rightarrow \max! \\
&M(u \mid v) \propto \exp(-(Hu-v)^2/2{\sigma_n}^2) \\
&N(\tau,A_1,A_2,\sigma_1,\sigma_2 \mid v) \overset{Separation}{\rightarrow} N_1(\tau),....,N_5(\sigma_2) \\
&N_i \propto x_i \exp(-x_i/\beta_i)
\end{aligned}
$$

<u>Results</u>

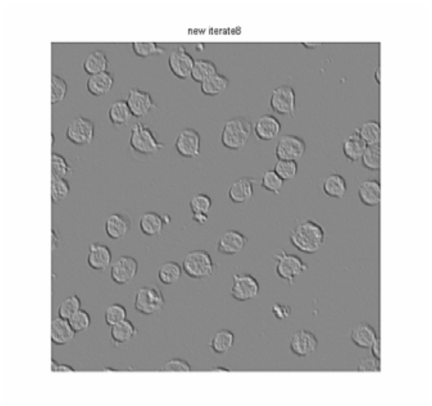
1.



2.



3.



1. Original DIC image
2. Result of iterative deconvolution and preconditioning, 20 iteration,
3. Result of blind deconvolution

HDFormGen: A Fast Nonlinear Approximation Formula Generator for Very High Dimensional Data Based on Variable Selection and Genetic Programming

Werner Groißböck

Department of Knowledge-Based Mathematical Systems
Fuzzy Logic Laboratorium Linz-Hagenberg
Johannes Kepler University Linz, A-4040 Linz, Austria
werner.groissboeck@jku.at

Summary. A new approach for finding nonlinear approximation formulas for very high-dimensional data is presented. This method has been developed for static data analysis, but it can be used for dynamic data analysis as well. The method is based on linear regression, but instead of the original variables we use nonlinear terms with these variables. Such a formula is still linear in the parameters, so ordinary least squares methods can be applied to find the globally optimal parameters. We use an accelerated version of genetic programming to find the optimal nonlinear terms, and we use variable selection methods to select those terms leading to an approximation formula which shows an optimal balance of accuracy and simplicity. In general, evolutionary methods like genetic programming tend to produce many individuals with low fitness. To save computation time, an early stopping strategy in case of low fitness is used. The method was tested with three benchmark data sets (the auto-mpg data set and the CPU data set in the UCI repository http://www.ics.uci.edu/ mlearn/MLRepository.html and the friedman data set in the KEEL repository http://sci2s.ugr.es/keel/). Although these data sets are only low dimensional and thus not in the core application area of our method, for the auto-mpg data set, an approximation formula has been determined, whose accuracy is comparable to the benchmark papers, for the CPU data set, an approximation formula has been achieved which is more exact than most of the benchmark papers, and for the friedman data set, an approximation formula has been determined which is more exact than all of the benchmark papers found so far.

1 Introduction

In the car industry, an engine test bench system is used which can measure up to 1500 variables. From time to time, some parts of the measurement system are in an invalid state, maybe because one of the sensors is overheated. To

safe time and money, such an invalid state has to be detected as soon as possible, and the experiment has to be aborted as soon as possible. So a system is needed, which can detect faults.
For most of the variables measured useful expert knowledge is not available. For this reason, only data driven methods can be used. Different methods are available. The major challenge is that the methods have to deal with a very high dimensionality.

The method HDFormGen (A fast nonlinear **For**mula **Gen**erator for **H**igh **D**imensional Data) can be used to find a nonlinear approximation formula for very high dimensional data. To demonstrate the strength of our approach, the following artificial data set with 201 variables and 800 entries has been constructed: The variables $x1, x2, ..., x200$ are filled with independent standard normally distributed numbers. The remaining variable (which we call y) is determined with the following formula:

$$\begin{aligned} y =& x1 \cdot (0.3 \cdot x5 - 0.6 \cdot (x3 \cdot x5 + x2 \cdot x6) \\ &+ 0.2 \cdot (x2 \cdot x4 \cdot x6 + x2 \cdot x3 \cdot x7 + x3 \cdot x4 \cdot x5 - x5 \cdot x6 \cdot x7)) \end{aligned} \tag{1}$$

We want to find an approximation formula for the variable y. [1] So we want to see if our only data driven method can find any reasonable results. After running our algorithm for half an hour (all our results have been processed on a 1600MHz pentium laptop) the following formula has been achieved:

$$\begin{aligned} y =& 9.4589e - 008 \\ &- 0.6 \cdot (x6 \cdot (x2 \cdot x1)) \\ &- 0.6 \cdot ((x3 \cdot x5) \cdot x1) \\ &+ 0.3 \cdot (x1 \cdot x5) \\ &+ 0.2 \cdot ((x7 \cdot x3) \cdot (x2 \cdot x1)) \\ &+ 0.2 \cdot (((x1 \cdot x4) \cdot x5) \cdot x3) \\ &+ 0.2 \cdot ((x6 \cdot x1) \cdot (x4 \cdot x2)) \\ &- 0.2 \cdot ((x1 \cdot (x5 \cdot x6)) \cdot x7) \end{aligned} \tag{2}$$

This formula is nearly identical to a simplified form of the formula in 1. The only difference is the constant $9.4589e - 008$, which is caused by the limitations of machine accuracy. The most important question is: Does the algorithm still work, when data sets containing noise have to be analyzed? To answer this question, the data set described above is used again, but now to each variable a certain amount of noise is added, before our algorithm is applied. As noise we use independent standard normally distributed numbers, which are divided by ten.

[1] The estimated standard deviation of y is 0.81222, so it is not zero, which would make the task trivial.

After an average time consumption of about 4.5 hours, the following approximation formula (for the noisy data set) can be achieved:

$$\begin{aligned} y =&0.0097468 \\ &- 0.56631 \cdot (x1 \cdot (x2 \cdot x6)) \\ &- 0.57815 \cdot ((x1 \cdot x3) \cdot x5) \\ &+ 0.28516 \cdot (x1 \cdot x5) \\ &+ 0.18876 \cdot (x2 \cdot ((x1 \cdot x7) \cdot x3)) \\ &+ 0.18276 \cdot (x1 \cdot (x5 \cdot (x3 \cdot x4))) \\ &- 0.17787 \cdot (x1 \cdot (x5 \cdot (x6 \cdot x7))) \\ &+ 0.18482 \cdot (((x4 \cdot x6) \cdot x1) \cdot x2) \end{aligned} \tag{3}$$

This formula is not identical to the formulas above. But if the subterms are compared, then we can see that all the subterms in formula 3 can also be found in formula 2 and vice versa. So the only real differences are the exact values of the parameters before each nonlinear subterm in the formulas. For example for the subterm with $x1, x3$ and $x5$, we get the parameter -0.57815 instead of the parameter -0.6. This slight modification of the parameters is a consequence of the noise that has been added to the data variables. If a data based method is used, and if you have to deal with noisy data, then a certain amount of error in the models achieved can never be avoided.
Conclusion: We have been able to find a formula that is 'nearly' equivalent to the formulas in 1 and 2. The only relevant differences are the real parameters in the formulas. For finding the correct parameters, we use the least squares algorithm, which finds the globally optimal parameter setting. Finally, the correct structure of the formula is found, and the globally optimal parameter setting!

2 The approximation formula generator HDFormGen

In this paper, the new algorithm *HDFormGen* (A **For**mula **Gen**erator for **H**igh **D**imensional Data) is introduced which is able to find an approximation formula with nonlinear terms for a high dimensional regression data set. With this algorithm, formulas similar to the following can been achieved:

$$y = \beta_0 + \beta_1 \cdot x1 \cdot x100 + \beta_2 \cdot \sin(x77) + \beta_3 \cdot \exp(x5/x6)$$

The basic idea of the algorithm is the following:

- The structure of each of the nonlinear terms in the whole formula is found and optimized with the use of genetic programming (see [5]).
- The parameters of the formula can be optimized easily with a least squares algorithm. This can only be done, if the formula is linear in the parameters, so the genetic programming tool must not generate terms which contain additional parameters.

There is another aspect that has to be considered:
The terms that are used in the approximation formula finally shall be as uncorrelated to each other as possible. We want an approximation formula which is on the one hand as simple as possible, and on the other hand as exact as possible. So we have to find the most important nonlinear terms, such that the regression formula based on these terms is as good as possible. Variable selection methods like *forward selection* have been designed to fulfill this task. In *HDFormGen* a variant of forward selection is used. For this reason, the basic concept of *forward selection* will be explained in the following rows:

- At first, the most important variable (or nonlinear term) is selected. This is that variable (or term) which is correlated strongest to the actual dependent y.
- Then the effects of the variables/terms selected so far are subtracted from the original dependent y. This is necessary to avoid that variables that are highly correlated to the first choice will be chosen again and again.
- Then, again the most important variable/term is selected.
- And again, the dependent is modified, such that the effects of the variables/terms chosen already are eliminated.
- Continue in this manner, until enough variables/terms are selected.

3 The new algorithm HDFormGen

3.1 The core of the new algorithm

In the following, the original dependent is called y. At the beginning, the actual dependent is the original dependent $y_{actual} = y$. Later y_{actual} will be modified. The constant term $c = (1, \ldots, 1)^T$ is always the first variable that is chosen. But this variable is not counted as real variable. The algorithm performs the following steps:

1. An accelerated version of genetic programming (including a population of individuals and a crossover operator) is used to generate millions of very simple formulas. We select that formula x_A which is best correlated with the actual dependent y_{actual}. We look only at the absolute value of the correlation coefficient.
2. Then we modify y_{actual} such that all the parts of y that can be approximated with the regressors already chosen are subtracted, setting y_{actual} to $y - \hat{y}(c, x_A)$. Here $\hat{y}(c, x_A)$ is the linear best approximation of y with the use of the regressors c and x_A. We can say, y_{actual} is y made orthogonal to the regressors already chosen.
3. Once again the accelerated version of genetic programming is used to generate millions of very simple formulas. And now we select that formula x_B which is correlated strongest with the actual dependent y_{actual}. We look only at absolute values again.

4. Then once again, y_{actual} is made orthogonal to the regressors already chosen, so we set y_{actual} to $y - \hat{y}(c, x_A, x_B)$.
5. Continue in this manner, until a given number of regressor terms is selected or some other termination criterion is fulfilled.

3.2 The accelerated version of genetic programming - an overview

Stopping the calculation of the correlation coefficient as early as possible, when it can be seen that the checked individual is not worth spending additional time, accelerates the algorithm enormously. But how can this be carried out, if we have a population of individuals and not a single individual? In the following lines the major steps of the accelerated genetic programming algorithm are described.

1. Generate an initial population with *popsize* individuals.
2. Evaluate each individual for $n1$ points of the training data set and estimate the correlation coefficient with the actual dependent by using only these $n1$ points.
3. Determine the $popsize_{small}$ best correlated individuals out of *popsize*, based on the estimated correlation coefficient. We look only at the absolute value of the correlation coefficient.
4. For these $popsize_{small}$ chosen individuals the *e*xact value of the fitness function (i.e. the absolute value of the correlation coefficient) using *a*ll the points of the training data set has to be calculated.
5. Produce a new generation of *popsize* out of the $popsize_{small}$ chosen individuals:
   - Repeat the following, until we have enough new individuals. Choose randomly two of the $popsize_{small}$ individuals and compare their fitness. The better one is called the winner, and the other one is called the loser. Let the winner produce two offsprings, one is an exact copy of the winner, and the other offspring is made via crossover (as crossover partner, one of the $popsize_{small}$ individuals is chosen, which is neither the winner nor the loser).
   - The individual which is the best so far is always copied into the next generation ('elitism').
   - A small part of the new generation is produced in the same way as the initial population. This is one way of avoiding the problem with local optima. A mutation is not needed any more.
6. Go to step 2, until a termination criterion is fulfilled.

- As termination criterion, we usually take that a specific number of generations is reached.
- The parameter *popsize* determines, how many individuals are evolved in the genetic programming algorithm. The parameter *popsize* can take any positive integer number. The larger *popsize* is, the more computation time

is needed, and the better the results are. In our experiments, a *popsize* of 5000 has been used successfully.

- The parameter $n1$ tells the algorithm, how many points are used to get a quick estimation of the correlation coefficient. $n1$ can be an arbitrary positive integer, but $n1$ shall not exceed the number of training data points. In our experiments, settings of $n1 = 30$, $n1 = 50$ and $n1 = 100$ have been used successfully.
- The parameter $popsize_{small}$ determines, how many individuals of the total population are selected to be examined in detail. The value of $popsize_{small}$ shall be much smaller than *popsize*, for example $popsize/10$.

4 Variants of the Formula Generator Algorithm Applied To Standard Benchmark Data Sets

4.1 The data set cpu

The data set 'cpu' can be found in the directory 'cpu-performance' of the UCI-repository, which can be found in the following address:

`http://www.ics.uci.edu/~mlearn/MLRepository.html`

Number of instances: 209
Number of attributes: 10
The data set contains the following attributes:

vendor name: string
model name: string
MYCT: machine cycle time in nanoseconds (integer)
MMIN: minimum main memory in kilobytes (integer)
MMAX: maximum main memory in kilobytes (integer)
CACH: cache memory in kilobytes (integer)
CHMIN: minimum channels in units (integer)
CHMAX: maximum channels in units (integer)
PRP: published relative performance (integer); the dependent variable;
ERP: estimated relative performance via linear regression (integer)

At first we deleted the attributes *vendorname* and *modelname* because our algorithm can not handle strings. Furthermore the data set contains the attribute *ERP*, which is an old estimation for *PRP*. So we have to delete the attribute *ERP*, because we do not want to generate an approximation formula by using the results of an old approximation. This would be too easy. So finally we have only 7 attributes remaining. Before the core of our algorithm has been run, we split the data into two parts: 70% of the 209 instances have been randomly chosen to play the role of the training data. And the other 30% play the role of the test-data.

Our algorithm has been started 10 times. Roughly 3.7 seconds are necessary per term for performing the evolutionary part of the algorithm. Totally we received ten approximation formulas, with an average MAE of 23.33 determined for the test data set. The worst MAE is only 25.15, and the best MAE is 23.06. The best formula is the following:

$$\begin{aligned} PRP =& 16.344 \\ & + 0.0032443 \cdot (sqrtabs((MMIN \cdot (MMAX \cdot CHMAX)))) \\ & + 0.7936 \cdot ((CACH - CHMAX) - \sin(CHMAX)) \end{aligned} \tag{4}$$

The MSE of this formula is 1394.9, and the RMSE is 37.348. In our standard benchmark paper (see [7]), various different methods have been tried out. The best method leads to an MAE of 38.0. So compared to this paper, our method leads to a more exact approximation.

Additionally, newer papers (see [10], [12], [2] and [1]) have been found, where the data set cpu is used.

Conclusion: In these papers, totally 30 variants of standard methods have been tried out. Only in 5 out of 30 cases, our approximation formula is outperformed.

4.2 The data set friedman

The data set 'friedman' can be found in the KEEL repository, in the following location:

http://sci2s.ugr.es/keel/datasets1.php?SID&codeds=36

In the keel repository, benchmark papers can be found. For the friedman data set, a quite actual (2004) benchmark paper is mentioned via the abbreviation 'Lee04' (see [6]).

We try to design our experiments as similar as possible to the benchmark paper, to get comparable results. In the benchmark paper, the following is done:

'This is a synthetic benchmark data set. Each sample consists of five inputs and one output. The formula for the data generation is $y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4^4 + 5x_5^5 + \varepsilon$, where ε is a Gaussian random noise $N(0,1)$, and $x_1, ..., x_5$ are uniformly distributed over the domain $[0,1]$. 1400 samples were created, of which 200 samples were randomly chosen for network training and 200 samples for validation. The remaining 1000 samples were used for network testing.'

In the KEEL repository, the data sets are already available as described in [6]. So we have a 200 sample training data set, and a 200 sample validation data set, and a 1000 sample test data set. Unlike the benchmark paper, we do not need any validation data. So we only take the 200 sample training data set to find an approximation formula, and we take the 1000 sample test data set to determine the quality. As a quality measure, here the MSE is used, according to the benchmark data.

Our formula detection algorithm has been run 20 times. Here we need 13 seconds for each term, and 30 second for finding the total formula, because the formula consists of two nonlinear terms, and four seconds are needed in the non-evolutionary part of the algorithm. The best formula that we get is the following:

$$\begin{aligned} out =& 4.8843 \\ &+ 10.1761 \cdot (in4 + \sin((in2 \cdot (in1 + (in1 + in1))))) \\ &- 5.3183 \cdot (\sin((in3 + (in3 + in3))) - in5) \end{aligned} \tag{5}$$

The MAE of this formula is 0.889281, the MSE of this formula is 1.23629, and the RMSE is 1.11189. In the benchmark-paper (see [6]), the best method leads to an MSE of 4.502. So our formula is much more exact.

Cross-validation and the data set friedman

For the dataset friedman, a tenfold cross validation experiment has been performed. For this experiment, a 1200-sample version of the friedman data set has been used, which can be downloaded from the KEEL repository, in the following location:

http://sci2s.ugr.es/keel/datasets1.php?SID&codeds=36

After the cross-validation, we have to calculate the average error measures on the test data files. We get an average MAE of 0.8394133, an average MSE of 1.1127881, and an average $RMSE$ of 1.0537323 .

So with cross validation, we finally get ten formulas. The formula, which reaches the best quality on the corresponding test data set, is the following formula:

$$\begin{aligned} out =& 4.9946 \\ &- 10.1215 \cdot (\sin(((in2 \cdot in1) \cdot (1.051813 \cdot -2.92026))) - in4) \\ &+ 20.4701 \cdot ((in3 \cdot in3) - ((-0.2465477 \cdot in5) + in3)) \\ &+ 2.9015 \cdot ((0.3611782 - (in1 \cdot in2)) \cdot (in1 \cdot \sin(in3))) \end{aligned} \tag{6}$$

This formula reached (on the test data set number 10) an MAE of 0.786382, an MSE of 0.963263, and an RMSE of 0.981459627 . The name of the corresponding test data file in the KEEL repository is 'Friedman-10-10tst.dat', so everybody is invited to check the quality of the formula! It has to be mentioned that here the best formula out of ten has been selected (via the test data), so we can not expected to get such a result in average. The average qualities have been stated above, and are more important.

Conclusion: For the friedman data set, all the benchmark papers that we found so far (see [3], [6] and [11]), have been outperformed by our method.

References

1. M. Birattari, G. Bontempi and H. Bersini, "Lazy Learning Meets the Recursive Least Squares Algorithm", MIT Press, Advances in Neural Information Processing Systems 11, Cambridge, 1999.
2. M. Ceci, A. Appice and D. Malerba, "Comparing Simplification Methods for Model Trees with Regression and Splitting Nodes", Springer Lecture Notes in Computer Science, Volume 2871/2003, ISBN: 3-540-20256-0, 2003.
3. Q. Fu, S. X. Hu, S. Y. Zhao, "Clustering-based selective neural network ensemble", Journal of Zhejiang University SCIENCE 2005 6A(5), ISSN 1009-3095, doi:10.1631/jzus.2005.A0387, 2005.
4. Hastie, T., Tibshirani, R., and Friedman J. , "The Elements of Statistical Learning: Data Mining, Inference, and Prediction.", Springer Berlin, 2001.
5. J. R. Koza, "Genetic Programming", The MIT Press, Cambridge, Massachusetts, 1992.
6. W. M. Lee, C. P. Lim, K. K. Yuen, S. M. Lo, "A Hybrid Neural Network Model for Noisy Data Regression", IEEE Transactions on Systems, Man and Cybernetics, Part B 34:2, Pages 951-960, 2004.
7. Merz, C. J., Pazzani, M. J., "Combining Neural Network Regression Estimates with Regularized Linear Weights", Advances in Neural Information Processing Systems 9, edited by M.C. Mozer, M.I. Jordan, and T. Petsche, 1997.
8. A. Miller, "Subset Selection in Regression - Second Edition", ISBN 1-58488-171-2 Chapman & Hall/CRC Boca Raton London New York Washington, D.C. 2002.
9. O. Nelles, "Nonlinear System Identification - From Classical Approaches to Neural Networks and Fuzzy Models", ISBN 3-540-67369-5 Springer-Verlag Berlin Heidelberg New York, 2001.
10. N. Rooney, D. W. Patterson, S. S. Anand and A. Tsymbal, "Random subspacing for regression ensembles", The Florida AI Research Society Conference, 2004.
11. D. P. Solomatine, M. B. L. A. Siek, "Flexible and Optimal M5 Model Trees with Applications to Flow Predictions", 6^{th} International Conference on Hydroinformatics, Liong, Phoon & Babovic (eds), ISBN 981-238-787-0 , World Scientific Publishing Company, 2004.
12. R. Setiono, W. K. Leow and J. Y. L. Thong, "Opening the neural network black box: an algorithm for extracting rules from function approximating artificial neural networks", Proceedings of the twenty first international conference on Information systems, Australia, 2000.

A note on ordinal sums of t-norms and t-subnorms on bounded lattices

Susanne Saminger
Department of Knowledge-Based Mathematical Systems
Johannes Kepler University
Linz, Austria
susanne.saminger@jku.at

Erich Peter Klement
Department of Knowledge-Based Mathematical Systems
Johannes Kepler University
Linz, Austria
ep.klement@jku.at

Radko Mesiar
Department of Mathematics and Descriptive Geometry
Slovak University of Technology
Bratislava, Slovakia
and
Institute for Research and Applications of Fuzzy Modeling
University of Ostrava
Czech Republic
mesiar@math.sk

Abstract — Ordinal sum t-norms on bounded lattices are discussed. The summand carriers are assumed to be subintervals of the bounded lattice and the summand operations are triangular (sub)norms.

Key words — *Triangular norm, Triangular subnorm, Ordinal sum, Bounded lattice.*

Accepted for the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2006), Paris

1 Introduction

Many-valued logics are usually based on a bounded lattice $(L, \leq, 0, 1)$ of truth values [7,8,16,20, 21]. In such a case, the conjunction is interpreted by some triangular norm on L. Most often L is chosen to be the unit interval and as such several principles for constructing t-norms on the unit interval are known like, e.g., additive and multiplicative generators or ordinal sums of t-norms and t-subnorms (for an overview on different methods see, e.g., [12, 13]). The latter ones are of particular importance, since, on the one hand, continuous t-norms (t-subnorms) on the unit interval are either the minimum or exactly ordinal sums of continuous Archimedean t-norms (and possibly, continuous Archimedean t-subnorms, see, e.g., [15,18,19]) and since, on the other hand, ordinal sums of t-subnorms represent the most general way to obtain a t-norm as an ordinal sum of semigroups as introduced by Clifford [9, 14].

Besides, lattices in general have been of interest as set of truth values such that triangular norms on these lattices have been investigated (see, e.g., [3–5, 10, 11, 25, 17, 22, 23]). Particularly in [22, 23], ordinal sum t-norms on bounded lattices L have been studied where the summands based on subintervals resp. sublattices of L have been t-norms on the corresponding sets only. It is remarkable that the demand for an arbitrary selection of the summand carriers which have to be pairwise disjoint except for their boundaries restricts the set of admissible bounded lattices to horizontal sums of chains. By this, the close relationship between the lattice structure itself and the possible choices for ordinal sum t-norms is exemplified. In this contribution, we will focus on ordinal sum t-norms on bounded lattices allowing also t-subnorms as summand operations on subintervals.

The following section provides the basic preliminaries and results on ordinal sums of posets and of semigroups, particularly of t-norms. Then we will turn to ordinal sum t-norms with t-norm summands on some bounded lattice and finally to ordinal sums on bounded lattices with t-subnorm summands as well.

2 Preliminaries

2.1 On some types of ordinal sums

Different types of ordinal sum concepts are used throughout the literature. One particular approach for building ordinal sums of disjoint posets is provided in [1] and can be generalized as follows:

Definition 2.1 Consider a linearly ordered index set $(I, \preceq_I)$, $I \neq \emptyset$ and a family of pairwise disjoint posets $(X_i, \leq_i)_{i \in I}$. The *ordinal sum* $\oplus_{i \in I} X_i$ (*in the sense of Birkhoff*) is defined as the set $\cup_{i \in I} X_i$ with the following order $\leq$ given by

$$x \leq y \Leftrightarrow (\exists i \in I : x, y \in X_i \wedge x \leq_i y) \text{ or } (\exists i, j \in I : x \in X_i \wedge y \in X_j \wedge i \prec_I j).$$

Note that by building ordinal sums of posets the preservation of the order of the underlying posets is guaranteed. Keeping this intention, we can relax the disjointness condition and introduce ordinal sums of intervals. Note that for any elements a, b of a poset $(X, \leq_X)$ with $a \leq_X b$, the interval $[a, b]$ is defined as $[a, b] = \{x \in X \mid a \leq_X x \leq_X b\}$ and we will denote by $]a, b[= [a, b] \setminus \{a, b\}$.

Definition 2.2 Consider a linearly ordered index set $(I, \preceq)$, $I \neq \emptyset$ and a family of intervals $([a_i, b_i])_{i\in I}$ such that for all $i, j \in I$ with $i \prec j$ either $[a_i, b_i]$ and $[a_j, b_j]$ are disjoint or $b_i = a_j$. The *ordinal sum* $\oplus_{i\in I}\, [a_i, b_i]$ is the set $\cup_{i\in I}\, [a_i, b_i]$ equipped with the order $\leq$ defined by

$$x \leq y \Leftrightarrow (\exists i \in I : x, y \in [a_i, b_i] \wedge x \leq_i y) \text{ or } (\exists i, j \in I : x \in [a_i, b_i] \wedge y \in [a_j, b_j] \wedge i \prec j).$$

In [2], ordinal sums have been introduced in the context of abstract semigroups in order to construct a new semigroup from a given family of semigroups. The basic idea is to extend an ordinally ordered system of non-overlapping semigroups into a single semigroup whose carrier is equal to the union of the original carriers.

Definition 2.3 [2] Let $(I, \preceq)$, $I \neq \emptyset$ be a linearly ordered index set, $(X_i)_{i\in I}$ a family of pairwise disjoint sets, and $(G_i)_{i\in I}$ with $G_i = (X_i, *_i)$ a family of semigroups. Put $X = \cup_{i\in I} X_i$ and define the binary operation $*$ on X by

$$x * y = \begin{cases} x *_i y & \text{if } (x, y) \in X_i \times X_i, \\ x & \text{if } (x, y) \in X_i \times X_j \text{ and } i \prec j, \\ y & \text{if } (x, y) \in X_i \times X_j \text{ and } i \succ j. \end{cases}$$

Then we say that $(X, *)$ is the *ordinal sum* of all $(X_i, *_i)_{i\in I}$. If necessary, we will refer to this type of ordinal sum as *ordinal sum in the sense of Clifford.*

Proposition 2.4 [2] *With all the assumptions of the previous definition the ordinal sum $(X, *)$ is also a semigroup, i.e., $*$ is an associative operation on X.*

Similar as in the case of ordinal sums in the sense of Birkhoff the condition of disjointness can be and has been relaxed for the case of ordinal sums in the sense of Clifford.

Proposition 2.5 [2] *Let $(I, \preceq)$, $I \neq \emptyset$ be a linearly ordered set, $(X_i)_{i\in I}$ a family of sets, and $(G_i)_{i\in I}$ with $G_i = (X_i, *_i)$ a family of semigroups.*

Assume that for all $i, j \in I$ with $i \prec j$ the sets X_i and X_j are either disjoint or that $X_i \cap X_j = \{x_{ij}\}$, where x_{ij} is both the unit element of G_i and the annihilator of G_j, and where for each $k \in I$ with $i \prec k \prec j$ we have $X_k = \{x_{ij}\}$. Put $X = \cup_{i\in I} X_i$ and define the binary operation $*$ on X by

$$x * y = \begin{cases} x *_i y, & \textit{if } (x, y) \in X_i \times X_i, \\ x, & \textit{if } (x, y) \in X_i \times X_j \textit{ and } i \prec j, \\ y, & \textit{if } (x, y) \in X_i \times X_j \textit{ and } i \succ j. \end{cases}$$

Then $(X, *)$ is a semigroup.

Note that ordinality in the sense of Clifford refers to the linear order of the index set I involved. The elements of some X_i, $i \in I$, need not fulfill some special order relation. On the other hand, taking into account that equality is an order relation on any set we immediately can state the following corollary.

Corollary 2.6 *Any ordinal sum in the sense of Clifford can be expressed as an associative operation on an ordinal sum of a family of sets in the sense of Birkhoff. Furthermore, it can be written as*

$$x * y = \begin{cases} x *_i y, & \text{if } (x,y) \in X_i^2, \\ x \wedge y, & \text{otherwise.} \end{cases}$$

with the corresponding conditions and notions from above.

In the case of t-norms on the real unit interval $[0,1]$ a variant of the construction of ordinal sums in the sense of Clifford was proposed (see, e.g., [6,13,15,24]).

Definition 2.7 Let $(]a_i,b_i[)_{i\in I}$ be a family of pairwise disjoint open subintervals of $[0,1]$ and let $(T_i)_{i\in I}$ be a family of t-norms. Then the *ordinal sum* $T = (\langle a_i,b_i,T_i\rangle)_{i\in I}\colon [0,1]^2 \to [0,1]$ is given by

$$T(x,y) = \begin{cases} a_i + (b_i - a_i)T_i(\frac{x-a_i}{b_i-a_i}, \frac{y-a_i}{b_i-a_i}), & \text{if } (x,y) \in [a_i,b_i]^2, \\ \min(x,y), & \text{otherwise.} \end{cases}$$

This type of ordinal sums is called *ordinal sum of t-norms*.

Note that ordinal sums of t-norms can be viewed as ordinal sums in the sense of Clifford (relaxing the disjointness requirement by allowing intervals overlapping in the boundary elements, and filling the gaps with the minimum). Therefore, associativity, monotonicity, commutativity and the neutral element are preserved by the construction process, and each ordinal sum of t-norms is again a t-norm.

2.2 Triangular norms and triangular subnorms on lattices

Consider a bounded lattice $(L,\wedge,\vee,0,1)$ with bottom element 0 and top element 1. A binary operation $T\colon L^2 \to L$ which is commutative, associative, non-decreasing in both arguments and has 1 as neutral element is called a *t-norm on* L (compare [4,22]). A binary operation $V\colon L^2 \to L$ which is commutative, associative, non-decreasing in both arguments and bounded from above by $\wedge$ is called a *t-subnorm on* L. Note that the structure of the lattice L heavily influences which and how many t-norms and t-subnorms on L can be defined. However, there exist, e.g., at least two t-norms for arbitrary bounded lattices L with $|L| > 2$, i.e., the minimum ${T_{\mathbf{M}}}^L(x,y) = x \wedge y$ and the drastic product

$${T_{\mathbf{D}}}^L = \begin{cases} x \wedge y, & \text{if } 1 \in \{x,y\}, \\ 0, & \text{otherwise,} \end{cases}$$

which are also the greatest and smallest possible t-norms on the corresponding lattice L.

3 Ordinal sums with t-norm summands

Before turning to ordinal sums involving t-subnorms as summands, let us first recall some results of ordinal sum t-norms where the summands are t-norms as well.

Definition 3.1 Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$ and some linearly ordered index set I. Further, let $(]a_i, b_i[)_{i\in I}$ be a family of pairwise disjoint subintervals of L and $(T_i)_{i\in I}$ a family of t-norms on the corresponding $[a_i, b_i]$. Then the *ordinal sum* $T = (\langle a_i, b_i, T_i\rangle)_{i\in I}\colon L^2 \to L$ is given by

$$T(x,y) = \begin{cases} T_i(x,y), & \text{if } (x,y) \in [a_i, b_i]^2\,, \\ x \wedge y, & \text{otherwise.} \end{cases} \tag{1}$$

Note that if $L = \oplus_{i\in I}\, [a_i, b_i]$, then an ordinal sum in the sense of Definition 3.1 is an ordinal sum in the sense of Clifford. But there also exist ordinal sum t-norms on lattices which are not an ordinal sum of intervals (see also [22, 23]). The following results provide necessary and sufficient conditions for ordinal sums defined by (1) being a t-norm.

Proposition 3.2 [22] *Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$, some linearly ordered index set $(I, \preceq)$, $I \neq \emptyset$ and a family of pairwise disjoint subintervals $(]a_i, b_i[)_{i\in I}$ of L. Then the following are equivalent:*

(i) The ordinal sum $T\colon L^2 \to L$ defined by (1) *is a t-norm for arbitrary T_i on $[a_i, b_i]$.*

(ii) For all $x \in L$ and for all $i \in I$ it holds that

(a) if x is incomparable to a_i, then it is incomparable to all $u \in [a_i, b_i[$,

(b) if x is incomparable to b_i, then it is incomparable to all $u \in]a_i, b_i]$.

How can one be sure that the selected family of pairwise disjoint subintervals $(]a_i, b_i[)_{i\in I}$ of a bounded lattice $(L, \wedge, \vee, 0, 1)$ is an admissible one for the construction of an ordinal sum t-norm? Note that the condition can be equivalently formulated as if for all $x \in L$ and all $i \in I$ there exists some $u \in]a_i, b_i[$ such that x is comparable to u, then x is also comparable to a_i and to b_i. This motivates the following illustrating result.

Proposition 3.3 *Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$, some linearly ordered index set $(I, \preceq)$, $I \neq \emptyset$, and a family of pairwise disjoint subintervals $(]a_i, b_i[)_{i\in I}$ of L. Denote by $\mathcal{C}_L$ the set of all maximal chains in L containing 0 and 1. Then the following are equivalent:*

(i) For all $C \in \mathcal{C}_L$ and for all $i \in I$ it holds that $C \cap]a_i, b_i[\neq \emptyset \Rightarrow \{a_i, b_i\} \subseteq C$.

(ii) For all $x \in L$ and for all $i \in I$ it holds that

(a) if x is incomparable to a_i, then it is incomparable to all $u \in [a_i, b_i[$,

(b) if x is incomparable to b_i, then it is incomparable to all $u \in]a_i, b_i]$.

Note that in case that $C \cap [a_i, b_i] \neq \emptyset$ it need not follow that $C \cap [a_i, b_i] = [a_i, b_i]$.

Now we look for conditions on the lattice L ensuring that, for arbitrary families of subintervals and arbitrary t-norm summands, the operation T defined by (1) is also a t-norm. Recall that a bounded poset $(X, \leq, 0, 1)$ is called a *horizontal sum* of bounded posets $((X_i, \leq_i, 0, 1))_{i\in I}$ if $X_i \cap X_j = \{0, 1\}$ whenever $i \neq j$, $X = \bigcup_{i\in I} X_i$, and $x \leq y$ if and only if there is an $i \in I$ such that $\{x, y\} \subseteq X_i$ and $x \leq_i y$.

Proposition 3.4 [22] *Consider some bounded lattice* $(L, \wedge, \vee, 0, 1)$. *Then the following are equivalent:*

(i) The ordinal sum T *as defined by* (1) *is a t-norm for arbitrary families of pairwise disjoint subintervals* $(]a_i, b_i[)_{i\in I}$ *and for arbitrary t-norms* $(T_i)_{i\in I}$ *on the corresponding* $[a_i, b_i]$.

(ii) $(L, \wedge, \vee, 0, 1)$ *is a horizontal sum of chains.*

In this case the set $\mathcal{C}_L$ of all maximal chains containing 0 and 1 coincides with the chains of the horizontal sum lattice. Moreover, each chain involved can be reconstructed by $C \cap L$ for some $C \in \mathcal{C}_L$.

4 Ordinal sums with t-subnorm summands

Let us now turn to ordinal sums which are built from t-subnorm summands.

Definition 4.1 Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$ and some linearly ordered index set I. Further, let $(]a_i, b_i[)_{i\in I}$ be a family of pairwise disjoint subintervals of L and $(V_i)_{i\in I}$ a family of t-subnorms on the corresponding $[a_i, b_i]$. Then the *ordinal sum of t-subnorms* $T = (\langle a_i, b_i, V_i\rangle)_{i\in I} \colon L^2 \to L$ is given by

$$T(x, y) = \begin{cases} V_i(x, y), & \text{if } (x, y) \in \,]a_i, b_i]^2\,, \\ x \wedge y, & \text{otherwise.} \end{cases} \tag{2}$$

In the sequel we will focus on conditions for the lattice itself, the family of chosen intervals and the family of corresponding summand operations for guaranteeing that the operation T defined by (2) is again a t-norm. Note that Definition 4.1 is not identical with Definition 3.1 since the domain where T acts as V_i differs in its boundaries. However, if all V_i are t-norms or if all subintervals $[a_i, b_i]$ are pairwise disjoint the definitions coincide. Further note that the ordinal sum of t-subnorms also differs from the notion of an ordinal sum in the sense of Clifford, since, e.g., $T|_{]a_i,b_i]^2}$ need not be a semigroup operation on $]a_i, b_i]$.

Lemma 4.2 Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$ and some linearly ordered index set I. Further, let $(]a_i, b_i[)_{i\in I}$ be a family of pairwise disjoint subintervals of L and $(V_i)_{i\in I}$ a family of t-subnorms on the corresponding $[a_i, b_i]$. Assume that the ordinal sum $T = (\langle a_i, b_i, V_i\rangle)_{i\in I} \colon L^2 \to L$ given by (2) is a t-norm, then for all $i \in I$, with $b_i = 1$ it holds that V_i is a t-norm.

4.1 Ordinal sums with one summand only

For simplicity reasons, we restrict our considerations first to an ordinal sum T on some bounded lattice $(L, \wedge, \vee, 0, 1)$ with one summand on some interval $[a, b]$ only. Since in case that $b = 1$, V on $[a, b]$ necessarily has to be a t-norm we assume that $b \neq 1$. The operation $T \colon L^2 \to L$ can then be written as

$$T(x, y) = \begin{cases} V(x, y), & \text{if } (x, y) \in \,]a, b]^2\,, \\ x \wedge y, & \text{otherwise.} \end{cases} \tag{3}$$

with V some t-subnorm on $[a, b]$ and $b \neq 1$.

Lemma 4.3 *Assume that $T\colon L^2 \to L$ defined by* (3) *is a t-norm for arbitrary V on $[a, b]$, $b \neq 1$. If some $x \in L$ is incomparable to a then it is incomparable to all $u \in [a, b]$.*

Therefore, the incomparability of some x to a implies also its incomparability to b. Vice versa, note that if some $x \in L$ is incomparable to b then it is incomparable to all $u \in]a, b]$ due to Prop. 3.2. However, it still might be comparable to a. In any case, the upper boundary b plays additionally a particular role when considering the ordinal sum with a t-subnorm summand.

Lemma 4.4 *Consider an interval $[a, b]$, $b \neq 1$ and a t-subnorm V on $[a, b]$. If $T\colon L^2 \to L$ defined by* (3) *is a t-norm, then either b is the neutral element of V, i.e., V is a t-norm, or for all $x, y \notin [a, b]$ it holds that $T(x, y) \neq b$.*

As a consequence, we can conclude the following for ordinal sums with arbitrary t-subnorm summands.

Corollary 4.5 *Consider an interval $[a, b]$, $b \neq 1$ and assume that $T\colon L^2 \to L$ defined by* (3) *is a t-norm for arbitrary t-subnorm V on $[a, b]$. Then for all $x, y \notin [a, b]$ it holds that $T(x, y) \notin]a, b]$.*

Finally, we achieve the following result providing necessary and sufficient conditions for an ordinal sum T being also a t-norm for arbitrary t-subnorm V on $[a, b]$.

Proposition 4.6 *Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$ and a subinterval $[a, b]$, $b \neq 1$. Then the following are equivalent:*

(i) The ordinal sum $T\colon L^2 \to L$ defined by (3) *is a t-norm for arbitrary t-subnorm V on $[a, b]$.*

(ii) For all $x \in L$ it holds that

(a) if x is incomparable to a, then it is incomparable to all $u \in [a, b]$,

(b) if x is incomparable to b, then it is incomparable to all $u \in]a, b]$,

and for all $x, y \notin [a, b]$ it follows that $T(x, y) \notin]a, b]$.

The conditions are rather restrictive, therefore, one might ask whether the situation relaxes for bounded lattices which can be represented as ordinal sums of intervals.

Lemma 4.7 *Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$ and a subinterval $[a, b]$, $b \neq 1$, such that $L = [0, a] \oplus [a, b] \oplus [b, 1]$. If there exists some $c \in L$, $c >_L b$ such that $L = [0, a] \oplus [a, b] \oplus [c, 1]$ then the ordinal sum $T\colon L^2 \to L$ defined by* (3) *is a t-norm for arbitrary t-subnorm V on $[a, b]$.*

Note that such a value c need not always exist although T on L is an ordinal sum of a t-subnorm on $[a, b]$, e.g., in case of $L = [0, 1]$ an arbitrary subinterval $[a, b]$ and an arbitrary t-subnorm V on $[a, b]$ can be chosen and still T defined by (3) is a t-norm.

As we have already seen by ordinal sums with t-norm summands not all lattices are appropriate candidates to fulfill the conditions of Prop. 3.2 for an arbitrary selection of the summand carriers as well as of the summand operations. Similar results hold in case of ordinal sums of t-subnorms.

Proposition 4.8 *Consider some bounded lattice $(L, \wedge, \vee, 0, 1)$. Then the following are equivalent:*

(i) For any $[a, b] \subseteq L$, $b \neq 1$, and any t-subnorm V on $[a, b]$ the ordinal sum $T\colon L^2 \to L$ defined by (3) *is a t-norm.*

(ii) L is a horizontal sum of chains.

4.2 More summands

Finally, we mention the results in case that several summands are taken into account. Note that the incomparability conditions as well as the property of T in case that its arguments are not from the same interval have to hold for all summand carriers. The structure of the lattice has again to be a horizontal sum of chains in case that both summand carriers as well as t-subnorms on these carriers are chosen arbitrarily.

Proposition 4.9 *Consider a bounded lattice* $(L, \wedge, \vee, 0, 1)$*, some linearly ordered index set* $(I, \preceq)$*,* $I \neq \emptyset$ *and a family of pairwise disjoint subintervals* $(]a_i, b_i[)_{i \in I}$*,* $b_i \neq 1$ *of* L*. Further, consider a family of t-subnorms* $(V_i)_{i \in I}$ *acting on the corresponding* $[a_i, b_i]$*. Then the following are equivalent:*

(i) The ordinal sum $T\colon L^2 \to L$ *defined by* (2) *is a t-norm for arbitrary t-subnorm* V_i *on* $[a_i, b_i]$*.*

(ii) For all $i \in I$ *and for all* $x \in L$ *it holds that*

  (a) if x *is incomparable to* a_i*, then it is incomparable to all* $u \in [a_i, b_i]$*,*

  (b) if x *is incomparable to* b_i*, then it is incomparable to all* $u \in]a_i, b_i]$*,*

  and for all $x, y \notin [a_i, b_i]$ *it follows that* $T(x, y) \notin]a_i, b_i]$*.*

Proposition 4.10 *Consider some bounded lattice* $(L, \wedge, \vee, 0, 1)$*. Then the following are equivalent:*

(i) The ordinal sum T *as defined by* (2) *is a t-norm for arbitrary families of pairwise disjoint subintervals* $(]a_i, b_i[)_{i \in I}$*,* $b_i \neq 1$ *and for arbitrary t-subnorms* $(V_i)_{i \in I}$ *on the corresponding* $[a_i, b_i]$*.*

(ii) L *is a horizontal sum of chains.*

5 Conclusion

We have investigated triangular norms on bounded lattices which are built as ordinal sums of t-norms as well as of t-subnorms. We showed in which way the possibility of choosing t-subnorms as summand operations further restricts the structure of the lattice, resp. demands further incomparability conditions for its elements and further restrictions on the operation itself. As in the case of ordinal sums with t-norm summands bounded lattices which are horizontal sums of chains only allow to choose the summand carriers as well as the summand operations arbitrarily.

Acknowledgements

Radko Mesiar gratefully acknowledges the support of VEGA1/3006/06 and VZ MSM 6198898701.

References

[1] G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, 1973.

[2] A. H. Clifford. Naturally totally ordered commutative semigroups. *Amer. J. Math.*, 76:631–646, 1954.

[3] B. De Baets and R. Mesiar. Triangular norms on product lattices. *Fuzzy Sets and Systems*, 104:61–75, 1999.

[4] G. De Cooman and E. E. Kerre. Order norms on bounded partially ordered sets. *J. Fuzzy Math.*, 2:281–310, 1994.

[5] G. Deschrijver and E. E. Kerre. Triangular norms and related operators in L^*-fuzzy set theory. In Klement and Mesiar [12], chapter 8, pages 231–259.

[6] S. Gottwald. *Fuzzy Sets and Fuzzy Logic. Foundations of Application—from a Mathematical Point of View*. Vieweg, Braunschweig/Wiesbaden, 1993.

[7] S. Gottwald. *A Treatise on Many-Valued Logic*. Studies in Logic and Computation. Research Studies Press, Baldock, 2001.

[8] P. Hájek. *Metamathematics of Fuzzy Logic*. Kluwer Academic Publishers, Dordrecht, 1998.

[9] S. Jenei. A note on the ordinal sum theorem and its consequence for the construction of triangular norms. *Fuzzy Sets and Systems*, 126:199–205, 2002.

[10] S. Jenei and B. De Baets. On the direct decomposability of t-norms over direct product lattices. *Fuzzy Sets and Systems*, 139:699–707, 2003.

[11] F. Karaçal and D. Khadjiev. $\vee$-Distributive and infinitely $\vee$-distributive t-norms on complete lattices. *Fuzzy Sets and Systems*, 151:341–352, 2005.

[12] E. P. Klement and R. Mesiar, editors. *Logical, Algebraic, Analytic, and Probabilistic Aspects of Triangular Norms*. Elsevier, Amsterdam, 2005.

[13] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*, volume 8 of *Trends in Logic. Studia Logica Library*. Kluwer Academic Publishers, Dordrecht, 2000.

[14] E. P. Klement, R. Mesiar, and E. Pap. Triangular norms as ordinal sums of semigroups in the sense of A. H. Clifford. *Semigroup Forum*, 65:71–82, 2002.

[15] C. M. Ling. Representation of associative functions. *Publ. Math. Debrecen*, 12:189–212, 1965.

[16] J. Łukasiewicz. Philosophische Bemerkungen zu mehrwertigen Systemen des Aussagenkalküls. *Comptes Rendus Séances Société des Sciences et Lettres Varsovie cl. III*, 23:51–77, 1930.

[17] G. Mayor and J. Torrens. Triangular norms on discrete settings. In Klement and Mesiar [12], chapter 7, pages 189–230.

[18] A. Mesiarová. Continuous triangular subnorms. *Fuzzy Sets and Systems*, 142:75–83, 2004.

[19] P. S. Mostert and A. L. Shields. On the structure of semi-groups on a compact manifold with boundary. *Ann. of Math., II. Ser.*, 65:117–143, 1957.

[20] V. Novák. *Fuzzy Sets and their Applications*. Adam Hilger, Bristol, 1989.

[21] E. L. Post. Introduction to a general theory of elementary propositions. *Amer. J. Math.*, 43:163–185, 1921.

[22] S. Saminger. On ordinal sums of triangular norms on bounded lattices. *Fuzzy Sets and Systems*. (accepted).

[23] S. Saminger, E.P. Klement, and R. Mesiar. On an extension of triangular norms on bounded lattices. (submitted).

[24] B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*. North-Holland, New York, 1983.

[25] D. Zhang. Triangular norms on partially ordered sets. *Fuzzy Sets and Systems*, 153(2):195–209, 2005.

The Machine Learning Framework for Mathematica: Creating interpretable, computational models from data

Thomas Natschläger, Mario Drobics, Felix Kossak
{thomas.natschlaeger, mario.drobics, felix.kossak}@scch.at

Software Competence Center Hagenberg, Austria
uni software plus GmbH, Linz, Austria

Abstract

In this presentation we will give an introduction to the machine learning framework for *Mathematica* (mlf). In particular we will give a general overview of mlf`s features and demonstrate the latest improvements in version 1.5.

Introduction

The *machine learning framework* for *Mathematica* is a collection of powerful machine learning algorithms integrated into a framework for the main purpose of data analysis. *Fuzzy logic* is one of its key techniques. The framework allows for combining different machine learning algorithms to solve one single problem. This combination of distinct algorithms may give the user unforeseen insights into its data. The algorithms are highly parameterizable. Given this parameterizability combined with the efficient core engine of the *machine learning framework* for *Mathematica*, the user is able to analyze their data interactively, with short cycles of changing parameter settings and examining the results.

The most common task in data analysis is to find relations of a set of input parameters to one or more output parameters. This kind of analysis is called *supervised*, as the goal of the analysis is given by the user. In the case that no explicit goal parameter is available, the analysis is called *unsupervised*. The *machine learning framework* provides algorithms for both kinds of tasks.

The *machine learning framework* for *Mathematica* combines
- efficiency and performance behind the scenes delivered by an optimized computational kernel - the core engine - realized in C++, and
- the ease of use supplied by the manipulation, descriptive programming, and graphical capabilities of *Mathematica*.

Powered by fuzzy logic

The framework draws its power from integrating *fuzzy logic* wherever possible. This yields results where crisp boolean yes/no decisions can be replaced with smooth, continuous transitions. In the realm of data analysis, smooth results very often model the underlying correlations within the data more realistically than crisp ones. For instance, the status of belonging to a certain class may be modeled more appropriately by allowing overlaps and degrees of membership instead of only the two possibilities of belonging or not belonging.

Modern software architecture

The software architecture of the *machine learning framework* for *Mathematica* is based on the principles of modern object-oriented and template-based programming. Standard libraries are used to ensure portability of the C++ kernel.

Mathematica front end

The *machine learning framework* for *Mathematica* is used from the *Mathematica* front end. By using the *Mathematica* front end, one has access to all *Mathematica* functions, including the graphical manipulation tools, and, with the *Mathematica* programming language, one has access to an elegant scripting language.

Wide range of machine learning algorithms

The *machine learning framework* for *Mathematica* covers a wide range of machine learning algorithms which can be integrated to work together and therefore yield new results.

Supervised Learning

- FS-ID3 (fuzzy decision trees)
- FS-LiRT (fuzzy regression trees)
- FS-FOIL (fuzzy rule learning)
- FS-MINER (fuzzy rule learning)
- LAPOC (optimization of fuzzy controllers / regression trees)
- RENO (optimization of fuzzy controllers)
- Ridge regression
- Additive regression
- Quadratic regression model
- Neuronal networks

Unsupervised Learning

- SOM (Kohonen maps)
- Fuzzy k-means (clustering)
- WARD clustering (crisp clustering)
- LVQ (learned vector quantisation)

Additional features

- Fuzzy logic (using different types of fuzzy sets and t-norms)
- Fuzzy inference (Mamdani, Sugeno, Tagaki-Sugeno-Kang)
- Advanced data visualisation and data manipulation
- Statistical methods (including correlation plots, mutual information with plots, etc)

Behind the *machine learning framework* for *Mathematica*

The *machine learning framework* for *Mathematica* is developed by Software Competence Center Hagenberg GmbH (SCCH) and is owned and distributed by uni software plus GmbH.
Software Competence Center Hagenberg GmbH, Hagenberg, Austria, is a company in the realm of the K*plus* programme, which was established by the Austrian government. This software was developed and is supported by the Knowledge Based Technology area of the Software Competence Center Hagenberg GmbH. uni software plus GmbH, Linz, Austria, has ten years of experience in distributing *Mathematica* and *Mathematica* based solutions. uni software plus GmbH collaborates with Wolfram Research, Inc., for worldwide distribution.