CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Dept. of Cybernetics

Bachelor's Thesis

Approximating by Fuzzy Conjunctions

Alexej Popovič
Cybernetics and Robotics

May 2020
Supervisor: prof. Ing. Mirko Navara, DrSc.

Acknowledgement / Declaration

I would like to thank my supervisor prof. Mirko Navara for his guidance and valuable advices he gave me during my work on this thesis.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 21st May 2020

.....

Abstrakt / Abstract

Tato práce se zabývá možnostmi aproximace dat fuzzy konjunkcemi, neboli trojúhelníkovými normami.

Cílem bylo zjistit, jaké metody jsou vhodné pro prokládání dat parametrizovatelnými trojúhelníkovými normami, jak charakterizovat kvalitu proložení a jak vstupní data ovlivní výslednou aproximaci.

Pro tyto účely byla nejdříve definována ztrátová funkce a poté byly uvedeny některé algoritmy pro optimalizaci této funkce. Byl použit jednoduchý algoritmus ternárního vyhledávání, ale i komplexnější metoda – optimalizace hejnem částic. Dále byly zavedeny znaky charakterizující kvalitu aproximace a nakonec byly zkoumány faktory ovlivňující výsledné proložení.

Dále byla diskutována alternativní metoda aproximace využívající generátory t-norem a byla představena možná vylepšení již existujícího postupu využívajícího této metody.

V závěru je naznačeno, jaké experimenty by bylo vhodné v budoucnu vykonat, aby bylo možné vybrat metodu aproximace vhodnou pro různé typy problémů.

Klíčová slova: Fuzzy konjunkce; aproximace, trojúhelníkové normy; t-normy; prokládání dat.

Překlad titulu: Aproximace fuzzy konjunkcemi

This thesis discusses ways of approximating data using fuzzy conjunctions (triangular norms).

The aim of the project was to find out which methods are feasible to fit data by parametric triangular norms, how to characterize the quality of the approximation and how the input data influence the result of the approximation.

For this, a loss function was defined and some algorithms for optimizing this function were introduced. The simple Ternary Search algorithm was used, as well as a more complex method of Particle Swarm Optimization. Furthermore, indicators characterizing the quality of the approximation were explored. Finally, the factors influencing the resulting approximation were discussed.

An alternative method of approximation using generators of t-norms was also discussed and improvements to an existing procedure based on this method were presented.

In the conclusion we suggest what experiments would be helpful to conduct in the future to help us choose the best technique for different types of problems.

Keywords: Fuzzy conjunction; approximation; triangular norms; t-norms; data fitting.

Contents

Tables / Figures

Chapter 1
Introduction

This thesis outlines basic techniques of approximating data using *fuzzy conjunctions*. These are usually modeled using triangular norms, although other options exist [2]. We will focus solely on triangular norms. Although the term *triangular norm* first appeared in the context of statistical metric spaces [10], they have spread into other branches of mathematics as well. They are being used in the fuzzy set theory and in fuzzy logic, where they serve as conjunctions; meanwhile their counterparts, triangular co-norms, are used for modeling disjunctions.

Approximating functions is one of the most deeply studied problems in science and it can be found in countless scientific fields. There are numerous techniques of regression analysis and function fitting. In general, the goal is to find a function $\hat{f}$ that represents the best approximation of another function f, i.e., $\hat{f} \approx f$. For this, we only have a finite set of values of the function, usually determined empirically.

Using fuzzy conjunctions for approximating data is an idea that is yet to be explored in depth, although the concept itself is simple and its potential could be great. One might need to create a fuzzy decision-making system based on some input data, or tune a fuzzy regulator to meet needed characteristics. In both cases the goal is to fit a given set of points by a triangular norm. As t-norms are by definition associative and thus are easily extended to take in more than two arguments, the input data could even be of higher number of dimensions.

The approximating t-norm can be one of the parametric families of t-norms which are determined by a parameter r, in which case we are trying to find the best value of the parameter for which the approximation is the most precise; alternatively we can approach the problem in a more general manner using the so-called additive generators, which are one-dimensional functions that completely determine a triangular norm. The first approach is very useful when we have a priori information about the input data source; for example which class of t-norms the approximated one belongs to. The second approach offers more flexibility in terms of the ability to generate an approximation of any possible t-norm.

Both approaches have their strengths and weaknesses and this thesis aims to describe and compare these techniques, while evaluating the robustness of the approximation, which points influenced the approximation the most and which data had a low or zero value for the approximation purposes.

Chapter 2
T-norms

Let us first recall the basic concepts of fuzzy logic. The notion of *many-valued logic* was originally defined by Jan Łukasiewicz in years 1913 [8] and 1920 [9]. He proposed a *three-valued logic*, which, besides true and false, also admits a third value of "possible". Later, Lotfi A. Zadeh introduced the fuzzy set theory in 1965 [17]. This theory deals with sets that have a "continuum of grades of membership". That means that elements can not only be or not be part of a set, they could also be a part of it just to "some extent". This, by extension, created the foundations for fuzzy logic, where the allowed truth values are not only true (1) or false (0), but anything in between, which means that it is an *infinite-valued logic*. To be practically usable, fuzzy logic needs to have operators dealing with membership values. The commonly used are the *Zadeh Operators* that implement conjunction (and, $\wedge$) using the `min` function, disjunction (or, $\vee$) using the `max` function and negation (not, $\neg$) by subtracting the input value from 1. These operations are computationally cheap and are usable in simple scenarios. However, *triangular norms* proved to be even more suitable for modeling conjunctions in fuzzy logic.

The first appearance of the term *triangular norm* (*t-norm* for short) dates back to 1942, when the paper by Karl Menger called *Statistical Metrics* [10] was published in a renowned American scientific journal.[1] The term was used to generalize the classical triangle inequality to be used in probabilistic metric spaces. T-norms garnered even bigger attention after Berthold Schweizer and Abe Sklar published *Espaces Métriques Aléatoires* in 1958 [14]. They offered a more complete set of axioms defining these operations which they also used in their later works [15–16]. The same axioms are still used as the definition of t-norms.

2.1 Definition

A *t-norm* is a binary operations generalizing logical conjunction in fuzzy logic [5]. It is a function $T : [0,1] \times [0,1] \mapsto [0,1]$ that satisfy the following axioms for all $x,\ y,\ z\ \in [0,1]$:

$$T(x,y) = T(y,x) \qquad \text{(commutativity)}$$

$$T(x,T(y,z)) = T(T(x,y),z) \qquad \text{(associativity)}$$

$$y \leq z \implies T(x,y) \leq T(x,z) \qquad \text{(monotonicity)}$$

$$T(x,1) = x \qquad \text{(neutrality of 1)}$$

Based on the definition, we can see that for all $x \in [0,1]$, all t-norms coincide on the boundary of the unit square. That is,

$$T(0,x) = T(0,x) = 0 \qquad \text{(commutativity + monotonicity)}$$

$$T(1,x) = T(x,1) = x \qquad \text{(commutativity + neutrality of 1)}$$

Notably, when putting 0 or 1 as the input arguments, the output of any t-norm is the same as if we used the boolean logic conjunction.

[1] Proceedings of the National Academy of Sciences of the United States of America

2.2 Basic types of t-norms

There are infinitely many t-norms as there are infinitely many functions satisfying the axioms. However, we do not need to discuss them all. Instead, we consider two groups of t-norms characterized by whether or not they are parametrized. Thus, we distinguish between the non-parametric t-norms and the parametric t-norms. Later on we will also discuss another way of defining t-norms, that is, using additive or multiplicative generators.

2.2.1 Non-parametric t-norms

The most basic types are those that are not parametrized. Some of these t-norms are the *minimum t-norm* T_{M} (also called the *Gödel t-norm*), the *product t-norm* T_{P}, the *Łukasiewicz t-norm* T_{L} and the *drastic t-norm* T_{D}:

$$T_{\mathrm{M}}(x, y) = \min(x, y)$$

$$T_{\mathrm{P}}(x, y) = x \cdot y$$

$$T_{\mathrm{L}}(x, y) = \max(x + y - 1, 0)$$

$$T_{\mathrm{D}}(x, y) = \begin{cases} x & \text{if } y = 1, \\ y & \text{if } x = 1, \\ 0 & \text{otherwise.} \end{cases}$$

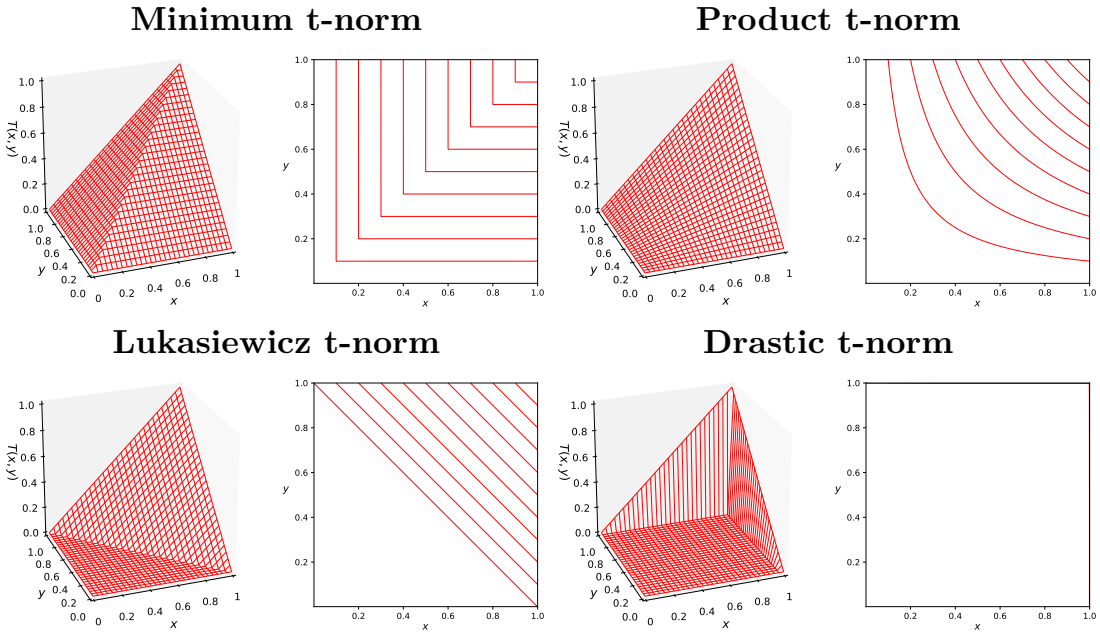

Figure 2.1. Surfaces and contours of the non-parametric t-norms

The minimum t-norm and the drastic t-norm represent the highest and lowest values, respectively, that any t-norm can have for all $x,\ y \in [0, 1]$.[1] That is, for every t-norm T and any values $x,\ y \in [0, 1]$ we have:

$$T_{\mathrm{D}}(x, y) \leq T(x, y) \leq T_{\mathrm{M}}(x, y)$$

[1] That is the reason why they are also called extremal.

2.2.2 Parametric t-norms

For the purposes of approximating data, parametric t-norms are more useful. Their values are determined not only by their arguments, but also by the value of their parameter r.[1] They are represented by so-called *families* of t-norms. Some of them are the *Frank t-norms* T_r^{F}, the *Hamacher t-norms* T_r^{H} and the *Yager t-norms* T_r^{Y}. They are given by [5]:

$$\left(T_r^{\mathrm{F}}(x,y)\right)_{r\in[0,\infty]} = \begin{cases} T_{\mathrm{M}}(x,y) & \text{if } r=0,\\ T_{\mathrm{P}}(x,y) & \text{if } r=1,\\ T_{\mathrm{L}}(x,y) & \text{if } r=\infty,\\ \log_r\left(1+\frac{(r^x-1)(r^y-1)}{r-1}\right) & \text{otherwise.}\end{cases}$$

$$\left(T_r^{\mathrm{H}}(x,y)\right)_{r\in[0,\infty]} = \begin{cases} T_{\mathrm{D}}(x,y) & \text{if } r=\infty,\\ 1, & \text{if } r=0 \text{ and } x=y=1\\ \frac{xy}{r+(1-r)(x+y-xy)} & \text{otherwise.}\end{cases}$$

$$\left(T_r^{\mathrm{Y}}(x,y)\right)_{r\in[0,\infty]} = \begin{cases} T_{\mathrm{D}}(x,y) & \text{if } r=0,\\ T_{\mathrm{M}}(x,y) & \text{if } r=\infty,\\ \max(0,1-((1-x)^r+(1-y)^r)^{1/r}) & \text{otherwise.}\end{cases}$$

Each of these t-norm families behave very differently in extreme values of their parameter. While the Frank t-norms approach the minimum t-norm with $r \to 0$, for r approaching 1 the limit of a Frank t-norm is equal to the product t-norm and with big values of r they converge to the Łukasiewicz t-norm. The Yager t-norms converge to the drastic t-norm for $r \to 0$, Łukasiewicz t-norm for $r \to 1$ and minimum t-norm for big r. Hamacher t-norms converge to the drastic t-norm for big r as well, but for small values of r they do not resemble any of the non-parametric t-norms.

Frank t-norm with $r = 0.1$ **Frank t-norm with $r = 1 \times 10^{12}$**

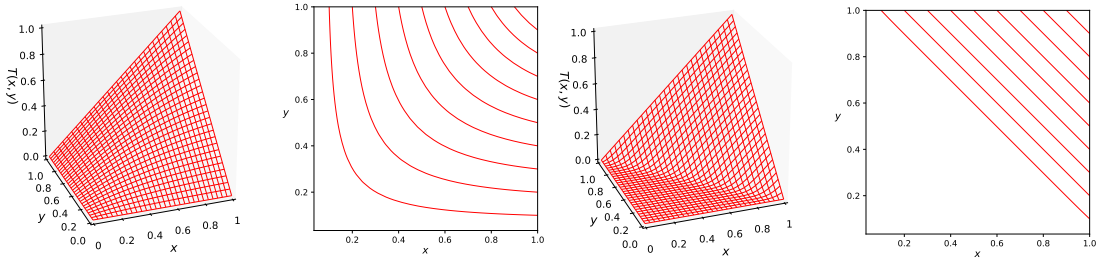

Figure 2.2. Surface and contours of the Frank t-norms

Hamacher t-norm with $r = 0.1$ **Hamacher t-norm with $r = 100$**

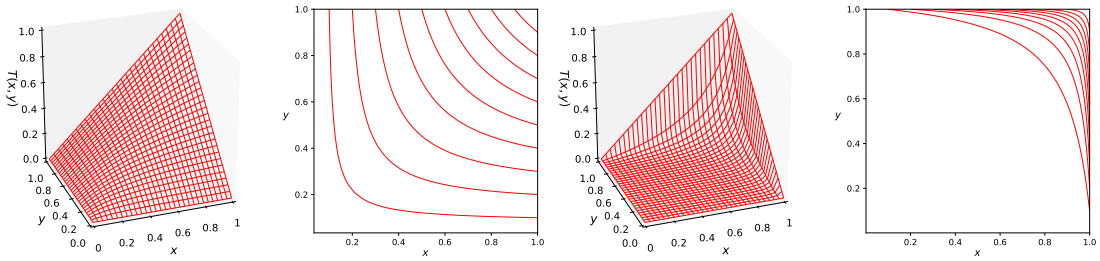

Figure 2.3. Surface and contours of the Hamacher t-norms

[1] More parameters are also possible, but we will consider only single-parameter t-norms.

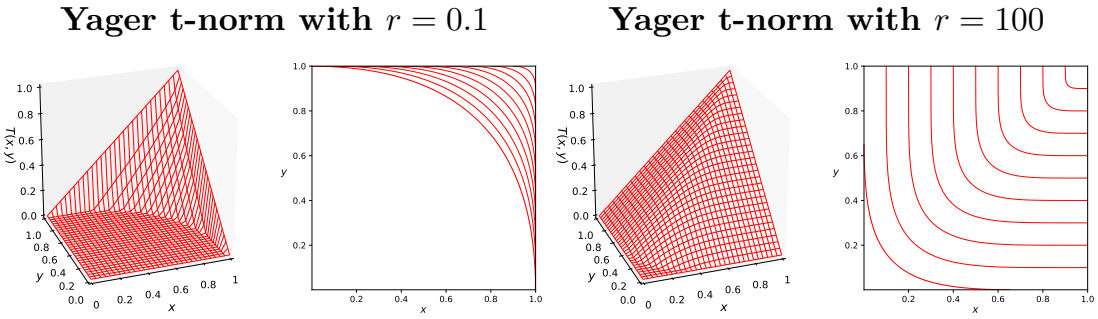

Figure 2.4. Surface and contours of the Yager t-norms

In this work we will not be considering the non-parametric t-norms to be used in approximation. Rather, we will be dealing only with the listed parametric t-norms and, later, also with the generator-based t-norms in Section 5.1. With the parametric t-norms, the optimization task lies in one-dimensional optimization of the parameter r, with respect to a chosen optimization criterion.

2.3 Classification and properties of t-norms

In this work, we shall use several important classes of t-norms. There are many ways to describe t-norms and to group them together, but for our purposes just a few of the most important notions and classes of t-norms are needed.

Idempotent values of a t-norm are those values x, for which $T(x,\ x) = x$. All t-norms have the so-called *trivial* idempotent elements 0 and 1. All idempotent values inside $]0, 1[$ are called *non-trivial*. Notice that all values $x \in [0, 1]$ are idempotent elements of T_{M}. In fact, due to the monotonicity requirement, T_{M} is the only t-norm for which this is true. Proof can be found in [5].

A continuous t-norm is called *Archimedean* if every sequence $x_n,\ n \in \mathbb{N}$, where $x_1 < 1$ and $x_{n+1} = T(x_n,\ x_n)$, converges to zero [11]. For example the product t-norm T_{P} is Archimedean, while the minimum t-norm T_{M} is not.

Thanks to the associativity property of triangular norms, we can also define an n-ary operation for each n-tuple $(x_1, x_2, \cdots, x_n) \in [0, 1]^n$ using induction [5]:

$$\mathop{\mathsf{T}}_{i=1}^{n} x_i = T\left(\mathop{\mathsf{T}}_{i=1}^{n-1} x_i,\ x_n\right)$$

If we have $x_1 = x_2 = \cdots x_n$, we can write:

$$x_T^{(n)} = T(x, x, \cdots, x)$$

This allows us to define Archimedean t-norms in a slightly different way, as it is defined in [5]: A t-norm T is called *Archimedean* if for each $(x, y) \in]0, 1[^2$ there is an $n \in \mathbb{N}$ with $x_T^{(n)} < y$. The two definitions are equivalent.

An element $x \in]0, 1[$, is called a *nilpotent element* of t-norm T if there exists some $n \in \mathbb{N}$ such that $x_T^{(n)} = 0$. All values of $x \in]0, 1[$ are nilpotent elements of the drastic t-norm T_{D}. On the other hand, T_{M} has no nilpotent elements.

An Archimedean t-norm T is *strict* if $T(x, x) > 0$ for all $x > 0$. Archimedean t-norms that are not strict are called *nilpotent*. This also means that strict t-norms have no nilpotent elements. We can see that while the Frank and Hamacher t-norms are strict, the Yager t-norm is not. Moreover, all values $x \in [0, 1[$ are nilpotent elements of T_r^{Y}.

Chapter 3
Data

3.1 Generating random data

While it would be beneficial to have real-world data for our experiments, the focus of this work is on approximating the data, not on modeling any real situations. In addition, we did not find any database dealing with approximation by triangular norms. That means we can generate the data on our own and later try to apply our methods on data obtained by observations or by conducting surveys.

Because of the monotony requirement which all t-norms must follow, we placed the same requirement on our random data that were going to be approximated by a t-norm. However, in reality the measured data might be subject to noise and/or measurement imprecisions, so small deviations from strict monotony could be considered. It appeared that this restriction is not trivial to follow and generating data that meet all our requirements is not a straightforward task.

First, we suppose that the desired values of a t-norm to be approximated are given in a rectangular grid $M \times M$, where $M = \{x_1, x_2, \ldots, x_n\} \subset]0, 1[$ and the sequence $(x_1, x_2, \ldots, x_n)$ is increasing. The desired value at node (x_i, x_j) will be denoted by $y_{i,j}$, $i, j \in \{1, \ldots, n\}$. Not all of the values need to be specified. This assumption can be used without loss of generality; sparse data can be always completed to a domain of this form. For simplicity, we denote $x_0 = 0$, $x_{n+1} = 1$. We do not assume the sequence $(x_0, x_1, x_2, \ldots, x_n, x_{n+1})$ to be equidistant, although we mostly use this special case.

One of the ways to generate such data could be generating a random associative copula as those are special continuous triangular norms [6, 12]. For that, we sample nonnegative numbers $r_{k,m}$, $k, m \in \{1, \ldots, n+1\}$, and put

$$y_{i,j} = \sum_{k=1}^{i} \sum_{m=1}^{j} r_{k,m} \,.$$

In order to satisfy the boundary conditions, we need

$$\begin{aligned} &\textstyle\sum_{k=1}^{n} \sum_{m=1}^{n} r_{k,m} = 1, \\ &\textstyle\sum_{k=1}^{i} \sum_{m=1}^{n} r_{k,m} = x_i \,, \qquad i = 1, \ldots, n \,, \\ &\textstyle\sum_{k=1}^{n} \sum_{m=1}^{j} r_{k,m} = x_j \,, \qquad j = 1, \ldots, n \,. \end{aligned}$$

The first condition can be easily satisfied by a normalization, but the others (containing it as a special case) make it a problem. We need a distribution over a complex polytope. If this distribution should be approximately uniform (or, at least, not very concentrated in a small part of it), the problem could be rather complicated. We did not proceed this way. Instead, we iteratively generated the values from $y_{n,n}$ to $y_{1,1}$ using the following algorithm.

3.1.1 Algorithm

1. Suppose the sequence $(x_0, x_1, x_2, \ldots, x_n, x_{n+1})$ is equidistant. As an example, let n be equal to 10. That means we get the following set of points: $M = \{0, 0.1, 0.2, \ldots, 0.8, 0.9, 1\}$. The complete set of grid points will then be $M \times M = \{(0,0), (0,0.1), (0,0.2), \ldots (1,0.8), (1,0.9), (1,1)\}$.
2. The values on the boundaries of the unit square are known as they are fixed for all t-norms. Next, we initialize the values of all nodes (x_i, x_j) for all $i, j \in \{1, \ldots, n\}$:

$$y_{i,j} = \begin{cases} \min(x_i,\ x_j), & \text{if } x_i = 1 \text{ or } x_j = 1, \\ 0 & \text{otherwise} \end{cases}$$

   The values that do not belong to the boundary will be changed in the next phase. We set them to zero here only for the purpose of having all the values initialized at the start of the algorithm.
3. Now iterate over the nodes that do not belong to the boundary. Beginning in the top right corner (x_n, x_n), traverse the diagonals going from left to right, top to bottom, as is visualized in Fig. 3.1, where the solid blue lines represent the sequential data generation while the red dashed lines represent moving to the next diagonal.
4. For each point $(x_i, x_j) \in M \times M$ generate a random number $p_{i,j}$ in the range $]0, 1]$ using a probability distribution of our choice. Now, to accommodate monotonicity, we need the value $y_{i,j} = (x_i, x_j)$ to be lower than both $y_{i+1,j}$ and $y_{i,j+1}$. We do that simply by putting $y_{i,j} = p_{i,j} \cdot \min(y_{i+1,j},\ y_{i,j+1})$.
5. Finish with the set $P = \{(x_1, x_1, y_{1,1}), \ldots, (x_n, x_n, y_{n,n})\}$.

This way of generating data ensures monotony, making the data usable for approximation by t-norms. At the same time, the data offer great variability – they can approach both the minimal t-norm, which represents the set of maximum values any t-norm can reach, and also the drastic t-norm, which represents the set of minimum values reachable by any t-norm.

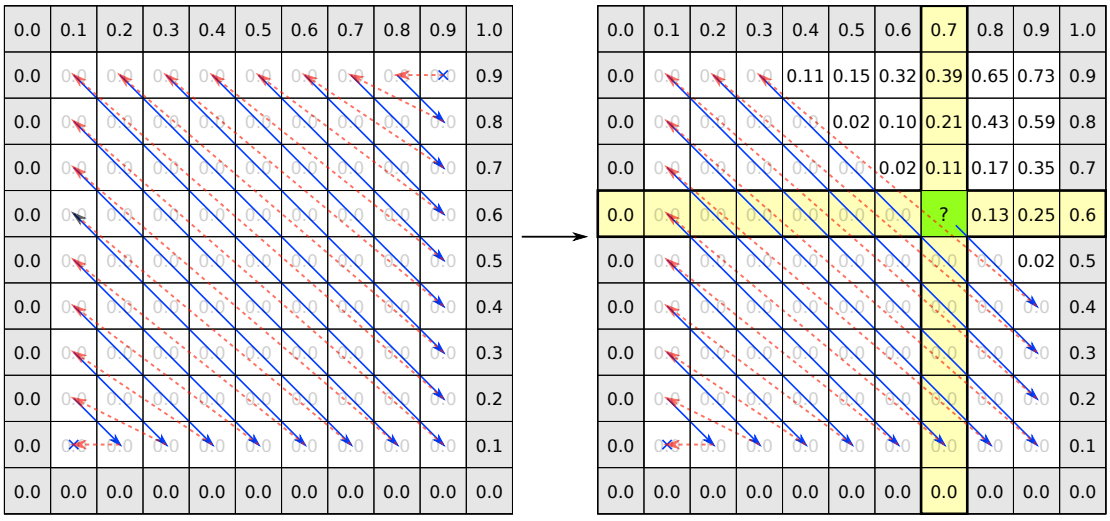

Figure 3.1. Process of data generation

An example of the data generation process can be seen in Fig. 3.1. After the grid initialization step, we iterate over diagonals of the inner area of the unit square. The green field in the figure will have a value of $p \cdot \min(0.11, 0.13) = p \cdot 0.11$.

To summarize the process of data generation using this algorithm – we proceeded from $y_{n,n}$ to $y_{1,1}$ as follows: We computed the upper bound required by monotonicity.

Then we multiplied this bound by a random factor from $]0, 1]$ and took this value. What remains is to choose the distribution in the random factor. If it was a proper constant (less than or equal to 1) and the nodes were equidistant, we would get the product t-norm scaled by this constant, which could be considered a desirable "prototype" of strict t-norms.

3.1.2 Choosing the probability distribution

We can generate the random number p in several ways by using different probability distributions. The decision of which distribution to choose largely depends on how the generated should look like. For example, when we have a priori knowledge about the input data, for instance which t-norm are they based on.

The easiest method of generating p is using the uniform distribution. However, that has some inconvenient drawbacks. The problem with using uniform distribution for picking p is that the values towards the upper right corner of the unit square have a lot greater influence over the subsequent values to be generated. That is because the values towards the lower left corner are always limited by the values already generated. By picking p from the uniform distribution independently of the previous values, the generated numbers are consequently pushed towards zero. In this essence, the values being generated are exponentially decreasing. This can be seen in Fig. 3.2. This grid of size 5x5 points represents mean values of 1 000 000 data sets. Notice how the values are very low already in the first few indices in the grid.

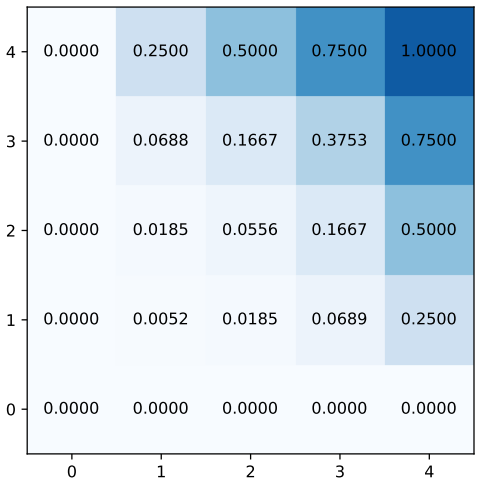

Figure 3.2. μ, Uniform distribution

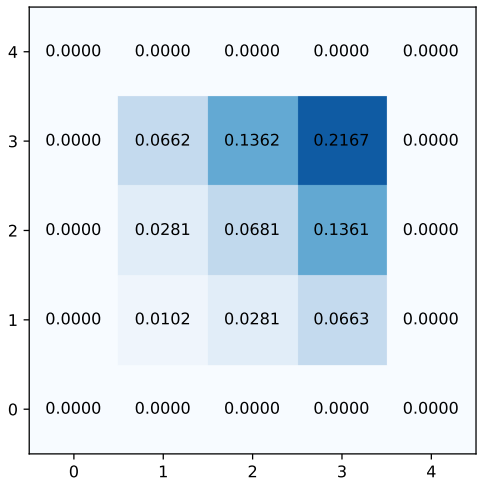

Figure 3.3. σ, Uniform distribution

An improvement can be made by taking the square root of p. That is, for each point (x_i, y_i), having m_i equal to the smallest non-zero number in the same column and n_i to the smallest non-zero number in the same row, the value at (x_i, y_i) will be equal to $z_i = \sqrt{p_i} \cdot \min(n_i, m_i))$.

As can be seen in Fig. 3.4, this gives us larger mean values. Also, the values in the center of the grid have greater standard deviations, see Fig. 3.5. This means that we get more diverse data, which is useful for testing our approximation techniques.

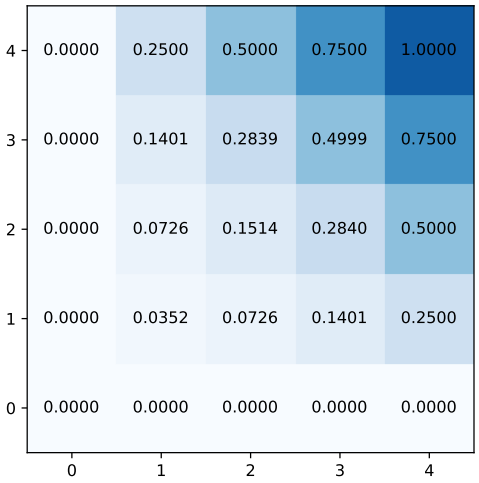

Figure 3.4. μ, Uniform distribution, using square root

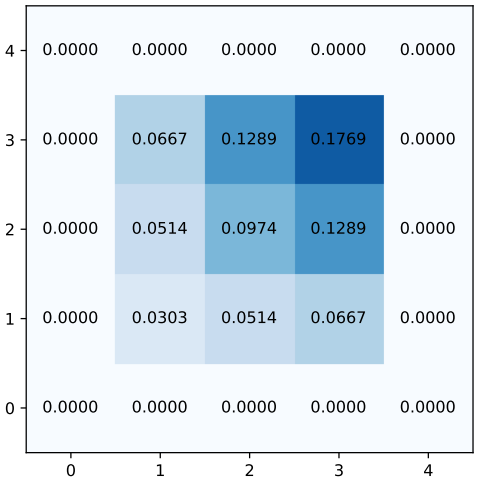

Figure 3.5. σ, Uniform distribution, using square root

Another approach is to use the *beta distribution*. The probability distribution function of the beta distribution for $x \in [0, 1]$ is defined as follows:

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)},$$

where B is the *beta function*:

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\mathrm{d}t$$

The integers α and β are parameters which can be tweaked to change the characteristics of the beta distribution. In our case it is best to shift the mean value during the algorithm so that on average we generate higher values of p for the upper right corner of the unit square and lower values for the lower left corner. That mitigates the problem of diminishing the values of the generated points. Here is the pseudocode for calculating p for a given point (x, y):

```
r = min(x, y)

a = a1 * r + a2 * (1 - r)
b = b1 * r + b2 * (1 - r)

p = beta(a, b)
```

Here, the values of a_1, a_2, b_1 and b_2 are parameters that are set before generating the data. They represent the start and end values of alpha and beta. That is, values that we need at the upper right corner of the unit square (where r will be high) and values needed at the lower left corner (where r will be low).

The parameters can be set to whichever values we need. For our experiments, these values served as a good starting point:

$$a_1 = 20, \quad a_2 = 1.5, \quad b_1 = 2, \quad b_2 = 1.5$$

These concrete values are only usable for the grid size of 5x5. They need to be tweaked in order to be used for generating data for grids of different sizes.

With these values we get the following distributions at $r = 1$, $r = 0.5$ and $r = 0$, see Fig. 3.6, 3.7 and 3.8. As you can see, we are essentially forcing the random variable

p to be greater when generating values in the upper right corner, while allowing small values in the opposite corner.

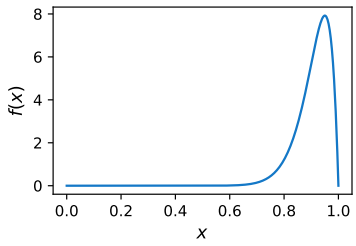

Figure 3.6. PDF, $r = 1$

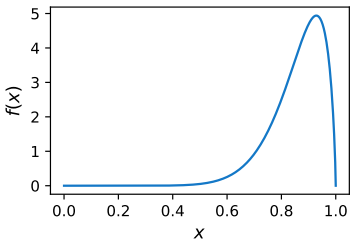

Figure 3.7. PDF, $r = 0.5$

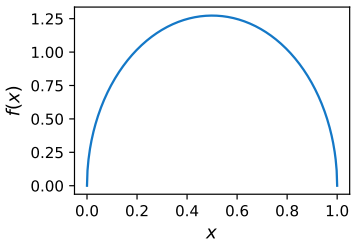

Figure 3.8. PDF, $r = 0$

When using the beta distribution, we get similar mean values and standard deviations to those obtained in the previous case, see Figures 3.9 and 3.10. However, now we have the option of modifying the parameters so that the characteristics of the generated data can be tweaked as needed.

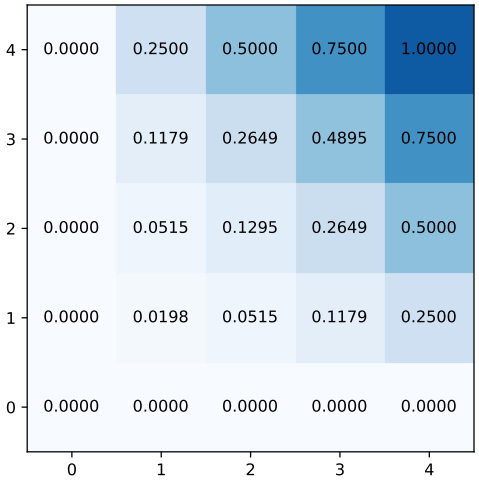

Figure 3.9. μ, Beta distribution

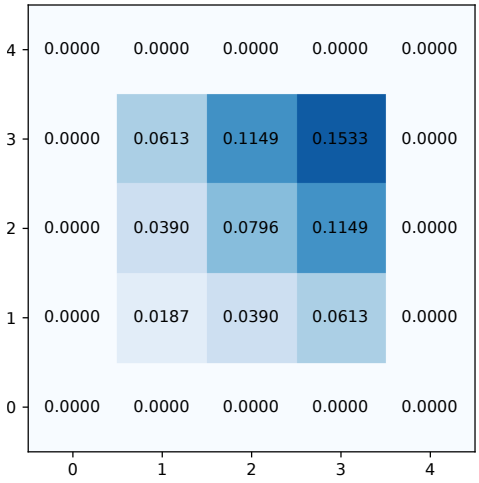

Figure 3.10. σ, Beta distribution

This procedure can also be interpreted so that each value was computed as a multiple of several samples of random factors (and bounds). In a logarithmic scale, this can be described as a sum of exponents, similar to the first method described above. There are three important differences. First, we start the computation from the point $(1, 1)$ (in practice, from (x_n, x_n)), instead of $(0, 0)$. Second, we do not sum all random exponents, only along one (maximal) path from $(1, 1)$ to the point in question. As a consequence, we do not have problems with the boundary conditions, which are easily implemented in the algorithm. Third, the first method generates values of some copula, which satisfies some restrictions (it has to be 1-Lipschitz). Not all t-norms satisfy this condition. Our method also is not restricted to 1-Lipschitz functions.

If the nodes are not equidistant, we can proceed the same way; generating $y_{i,j}$, we just multiply the random exponent by a scaling factor to reflect the distance from the previous points, e.g. by the length of the respective interval, $x_{i+1} - x_i$ or $x_{j+1} - x_j$, depending on which bound is active. We could also generate a dataset with equidistant points and only take a subset (perhaps randomly) of it which would create a dataset that is less regular and with nodes that have varying distance from each other.

3.1.3 Non-monotonicity

It is not strictly required to have the input data monotonic. In practice, the gathered data could be noisy or could be a result of more complex computations than merely a

fuzzy conjunction. Or, when the data are human-based, e.g. collected through surveys, they might not be monotonic simply because humans are not always very precise when thinking abstractly.

Our method could be easily extended to yield also non-monotonic data, for example by adding random noise to the final generated set of points. For our experiments, however, we did not do that.

3.1.4 Non-commutativity

The data can violate commutativity. In such a case, we can have $y_{i,j} \neq y_{j,i}$, although each t-norm has the same value at (x_i, x_j) and (x_j, x_i). So we have two different desired data items for a single value. This need not be a problem; in approximation, we can also admit more values at the same node. The loss function that we define later in Section 4.1 is not influenced by this. If we really needed to, we could modify the data to be monotonic while having no influence over the approximation. For this, we can use the symmetry of all t-norms – their commutativity. We can freely move any point to the opposite side of the unit square across the diagonal simply by switching its coordinates x_i and x_j. That is, approximating points $(x_i, x_j, y_{i,j})$ and $(x_j, x_i, y_{j,i})$ would be equal to approximating two points $(x_i, x_j, y_{i,j})$ and $(x_i, x_j, y_{j,i})$. So, to achieve commutativity in our generated data, we could simply create a copy of each point with the switched coordinates. That is, having created a dataset P_1, we would create a new set of points $P_2 = \{(x_j, x_i, y_{i,j}) \mid (x_i, x_j, y_{i,j}) \in P_1\}$. The final set would be equal to $P = P_1 \cup P_2$.

Another method of generating data satisfying commutativity, and perhaps the simplest, could be generating the data only in one half of the unit square, that is, points $y_{i,j}$ for all $i, j \in \{1, \ldots, n\}, i \leq j$ and then copying tha values to the other half, or rewriting the generated values by the average of the two opposing values on each side of the unit square's diagonal, that is, $y_{i,j} = (y_{i,j} + y_{j,i})/2$ and then $y_{j,i} = y_{i,j}$.

Chapter 4
Optimization of parametric t-norms

4.1 Criteria for optimization

There are several options for choosing optimization criteria. The criterion that we used is the sum of squares of the error terms for each point in the generated set. Given a point (x, y, z) and a t-norm T_r, an error term is calculated as the difference of the z value and the value of the t-norm at (x, y).

$$l_{T_r}(P) = \sum_{(x,y,z)\in P} (z - T_r(x, y))^2 \tag{1}$$

As can be seen in (1), the optimization method is in fact the well known Least Squares method:

$$r^* = \operatorname*{argmin}_r l_{T_r}(P) = \operatorname*{argmin}_r \sum_{(x,y,z)\in P} (z - T_r(x, y))^2$$

Let us denote the value of the optimization criteria for the optimal value of r by $l^*_T(P)$.

4.2 Algorithm 1 – Ternary Search and Golden Section Search algorithms

One of the simplest approaches to finding the global minimum of a function is using the *Ternary Search*[1] algorithm. The algorithm operates with a function that is convex and unimodal on an interval $[a, b]$. This means that the function has only a single minimum on the interval $[a, b]$. The need for unimodality, which arises from the algorithm principle, usually does not represent a significant problem. However, in some cases the function of optimization criteria w.r.t. parameter r is not convex and unimodal on the whole interval. Namely, the Yager t-norms are nilpotent, causing an area of values that are exactly zero. Data in this area cannot be optimized due to the fact that the change of the parameter r does not induce a change in the error terms connected with this data. This leads to the loss function having multiple minima on the interval $[a, b]$ and the Ternary Search algorithm might (and will) have problems finding the global minimum.

Parametric t-norms generally operate with values of r in the interval $[0, \infty)$, which is not feasible in real world situations. In real use, it is sufficient to work with a closed interval. During the first phase of the algorithm, we find an interval sufficiently large so that it contains the minimum. In the next phases we locate the minimum.

[1] Note that there is also an algorithm for finding a certain element in a sorted array with the same name. Here we mean the algorithm for finding an extreme value of an unimodal function inside a given interval.

The steps of the algorithm are as follows:

1. Begin with an interval $[a_0, b_0] = [1 \cdot 10^{-15}, 10]$. Suppose that $r^* > a_0$. Thus, $l(r^*) < l(a_0)$. We need to find an interval $[a_0, b_0]$ such that $l(r^*) < l(a_0)$, $l(r^*) < l(b_0)$ and $r^* \in [a_0, b_0]$, meaning that the function l is decreasing on $[a_0, r^*]$ and increasing on $[r^*, b_0]$. We can do that by moving the right bound until the value of $l(b_0)$ starts to rise. That is, update the value $b_0 = 2 \cdot b_0$ until $l(b_0) \geq l(2 \cdot b_0)$. Now we know that $r^* < 2 \cdot b_0$. So, we multiply b_0 by 2 once more so that the optimal value lies inside the interval $[a_0, b_0]$. This will be the interval $[a, b]$ we will be searching in.
2. Start searching with an interval $[a, b]$, assuming the minimum of the function $l(r)$ is located inside this interval. In this interval, choose two points m_1 and m_2. There are several ways of choosing these points and they have an impact on the convergence rate of the algorithm. Let us pick them by dividing the interval to three segments of the same size[1]:

$$m_1 = a + \frac{b-a}{3}, \quad m_2 = b - \frac{b-a}{3} \tag{2}$$

   Consequently, the size of the interval $[m_1, m_2]$ will decrease in each iteration by a factor of $2/3 \approx 0.667$ in the worst-case scenario, i.e., when the optimal value never lies in the center of the interval.

   Other algorithms based on different options of choosing these points exist. For example the *Golden Section Search* [13], which uses the following formula for choosing the next point:

$$m_1 = a + (b-a) \cdot \frac{3-\sqrt{5}}{2}, \quad m_2 = b - (b-a) \cdot \frac{3-\sqrt{5}}{2}$$

   Here, the size of the interval $[m_1, m_2]$ decreases by a factor of $(\sqrt{5}-1)/2 \approx 0.618$ in the worst-case scenario. We can see a slight improvement here and indeed, this algorithm proves to be optimal when the extreme is sampled from the uniform distribution.

   Another option might be the *Fibonacci Search*[2], which makes the intervals $[a, m_2]$ and $[m_1, b]$ to have the same ratio as two consecutive Fibonacci numbers.

   Both the Golden Section Search and the Fibonacci Search algorithms were discovered by Jack Kiefer in 1953 [4].
3. One of the following conditions will apply for the points m_1 and m_2:
   - $l(m_1) > l(m_2)$ – r^* cannot be lower than m_1, therefore it must be inside the interval $[m_1, b]$. The next iteration will operate on the interval $[a, b] = [m_1, b]$.
   - $l(m_1) < l(m_2)$ – r^* must be lower than m_2, therefore must lie in the interval $[a, m_2]$. The search interval for the next iteration will be $[a, b] = [a, m_2]$.
   - $l(m_1) = l(m_2)$ – points m_1 and m_2 either lie inside an interval, where the function $l(r)$ is constant[3], or they have mapped to the same value by chance and the minimum will be between these points. In either case, for the next iteration we have $[a, b] = [m_1, m_2]$.
4. After obtaining a new interval for searching of the minimum, we first check the width of this interval $d = b - a$. If it is lower than a predefined value (arbitrarily small positive number ε), we can assume that, up to some accuracy, the minimum

[1] Hence the name Ternary Search
[2] Not to be confused with the same named algorithm for searching in a sorted array
[3] This can occur with nilpotent t-norms

corresponds to any point inside this interval. Without loss of generality, we can declare that we have found a sufficiently good approximation of the optimum $r^* = a$. Similarly, if we reach a predefined limit of iterations, the algorithm is terminated as well. If any of the terminating conditions is not met, we continue with the next iteration with the new interval $[a, b]$.

While being relatively fast and efficient, the Ternary Search algorithm and other algorithms of this type are not usable for multimodal optimization as they could converge into a local minimum. For those we need to use other techniques.

4.3 Algorithm 2 – Particle Swarm Optimization

As previously stated, in some cases we might have to find the global minimum of a function on an interval on which the function is not unimodal. This is also the case of the loss function l_r for the Yager t-norm. The precise explanation of why this happens will be given in chapter 4.4.

Suppose we are given the following loss function:

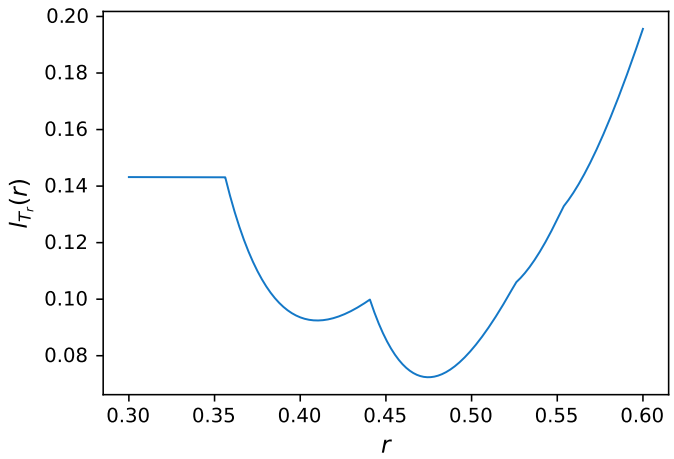

Figure 4.1. Multimodal loss function

As we can see, the function has two local strict minima at $r_1 = 0.410$ and $r_2 = 0.475$, which is also the global minimum. One of the ways to find global extrema on multimodal functions is to use one of the population-based strategies, which operate with multiple possible solutions at once. One of such techniques is called *Particle Swarm Optimization*, or *PSO* for short.

The algorithm was introduced by J. Kennedy and R. Eberhart in 1995 [3]. It is a population-based, stochastic optimization algorithm that was inspired by the behavior of bird flocks and fish schools. The algorithm is initialized with a population of random solutions, called particles, that are updated in each step. In this regard, it is similar to genetic algorithms. However, it does not use any evolution operators to combine or modify the solutions. Instead, the particles fly through the problem space and are drawn to the currently best solution.

The algorithm here is a slightly modified version of the one given by Kennedy and Eberhart. In general case, the algorithm can be used for multi-dimensional optimization. Here we are optimizing the loss function $l(r)$, which is one-dimensional.

Each particle i has its current position $p_{i,t}$ and velocity $v_{i,t}$ at iteration (time) t, its best known position $p^*_{i,t}$ and best known value $l^*_{i,t}$. It also knows the position of the globally best particle P^*_t and its value L^*_t.

The steps of the algorithm are as follows:

1. Create a population of N particles with their initial positions uniformly distributed over the maximum interval of r so that the distance between two neighboring particles is constant. The interval depends on the family of t-norms we are optimizing. For example, the starting interval for the Yager t-norms could be $[1 \cdot 10^{-15}, 100]$, which gives us enough range. In fact, higher values of r than 100 are problematic for the Yager t-norm because of numerical instability. For strict t-norms, the interval can be found iteratively in the same way as in the previous algorithm.
2. At each iteration t, do the following:

   - For each particle i, update its current position and the velocity for the next iteration. The current position equals the previous position plus current velocity:

     $$p_{i,t} = p_{i,t-1} + v_{i,t}$$

     Randomly sample two numbers r_1 and r_2 in range $[0, 1]$ using the uniform distribution. The new velocity will be equal to the current velocity plus the vector[1] towards its own best known position plus the vector towards the globally best position:

     $$v_{i,t+1} = v_{i,t} + 2r_1(p^*_{i,t} - p_{i,t}) + 2r_2(P^*_t - p_{i,t})$$

     If the new value of the loss function of the particle is lower than its best known value, update the particle's best known position and value:

     $$p^*_{i,t+1} = \begin{cases} p_{i,t}, & \text{if } l_{i,t} < l^*_{i,t}, \\ p^*_{i,t} & \text{otherwise} \end{cases}$$

     $$l^*_{i,t+1} = \begin{cases} l_{i,t}, & \text{if } l_{i,t} < l^*_{i,t}, \\ l^*_{i,t} & \text{otherwise} \end{cases}$$

   - Find the particle m with lowest l_t and update the globally best value and position:

     $$L^*_{t+1} = l_{m,t}$$

     $$P^*_{t+1} = p_{m,t}$$

   - After updating the globally best value and position, find the particle m with the greatest difference of its value from the globally best value. If this difference is smaller than a predefined value (arbitrarily small positive number ε), we can assume that up to some accuracy the minimum corresponds to any of the particles. Without loss of generality, we can assume that we have found a sufficiently good approximation of the optimum $r^* = P^*_t$. If we reach a predefined limit of iterations, the algorithm is also terminated. If any of these conditions is not met, we continue with the next iteration $t + 1$.

The progress of the optimization can be seen in the following figures. In Fig. 4.2 we can see the initialization step, where the population is uniformly distributed over the input function. Then the particles start to gather around the global optima, where they settle around iteration 80, as can be seen in Fig. 4.5.

[1] In our case, the vector is one-dimensional. In general case, it can have more than one dimension.

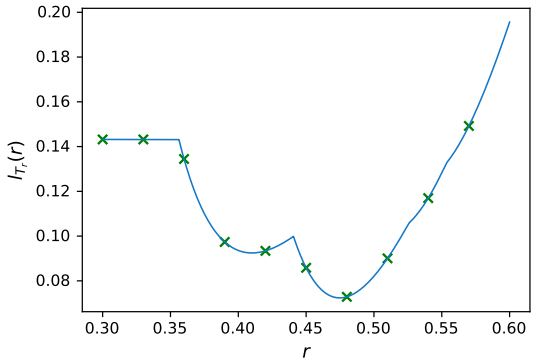

Figure 4.2. PSO initialization

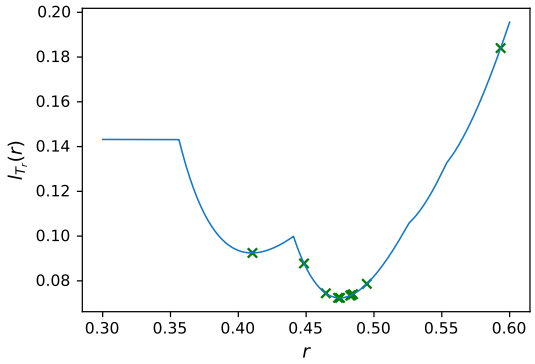

Figure 4.3. PSO at iteration 10

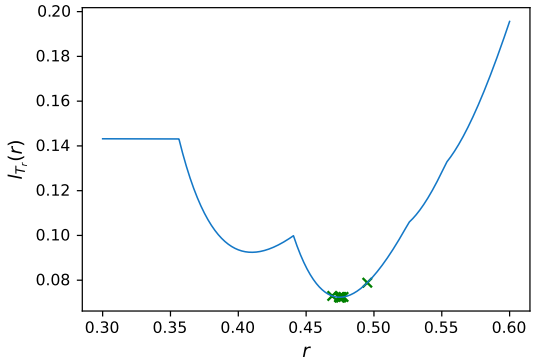

Figure 4.4. PSO at iteration 40

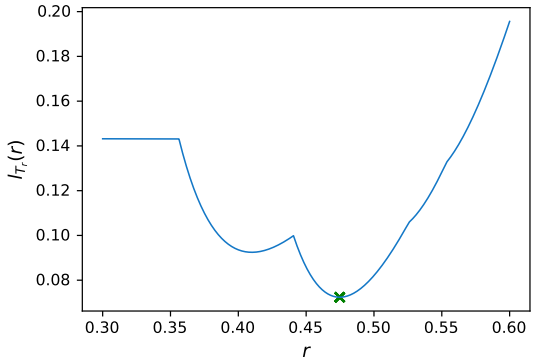

Figure 4.5. PSO at iteration 80

4.4 Measure of quality of approximation

It turns out that in real world applications, the loss function $l(P)$ is not enough to give us full understanding of how well a t-norm approximates some data. This is because the data may have been obtained with uncertainty, the measurements may have been imprecise, and then we would need to know the uncertainty of the found optimum. Or, the data might be incomplete, requiring that new points be added to our input data. Consequently, we need to know how well the t-norm is able to adapt to these new circumstances.

4.4.1 Tolerance interval

Let us define *tolerance interval* as an interval of values r, for which the ratio of the loss function $l_{T_r}(P)$ and the optimal loss $l_T^*(P)$ is less than or equal to a predefined value $k+1$:

$$S_T(P,k) = [r_1, r_2] \;\mid\; \forall r \in [r_1, r_2] : \frac{l_{T_r}(P)}{l_T^*(P)} < k+1$$

4.4.2 Integral of a t-norm over tolerance interval

To see how much a given t-norm changes with values $r \in S_T$, we can take its integral over the unit square and over the tolerance interval. Assuming that the family of t-norms depends monotonically on the parameter, which is the case of all families considered here, we can write:

$$I_S = \int_0^1 \int_0^1 \left| \int_{r_1}^{r_2} T_r(x,y)\ \mathrm{d}r\,\mathrm{d}x\,\mathrm{d}y \right| = \iint_A \left| \int_{r_1}^{r_2} T_r(x,y)\ \mathrm{d}r\,\mathrm{d}x\,\mathrm{d}y \right|,$$

where A is the area of the unit square. Of course, this can also be written as:

$$I_S = \iint_A |T_{r_2}(x,y) - T_{r_1}(x,y)|\,\mathrm{d}x\,\mathrm{d}y$$

The value of I_S gives us a measure of how much the t-norm has to *flex* in order to have a specific deviation from T_{r^*}. What does it tell us? Let us demonstrate that on an example using the Hamacher t-norms.

We will show what happens when we try to modify data that are concentrated around *the center of the unit square* and when the data are closer to *the edge of the unit square*. We will denote these two cases by indices A and B.

4.4.3 Example – case A

In the first case, consider having input data like this:

$$P_A = \{(0.4, 0.4, 0.02), (0.6, 0.4, 0.07), (0.4, 0.6, 0.3), (0.6, 0.6, 0.5)\}$$

After fitting these points, we get the following values for the optimal parameter and the respective loss and tolerance interval ($k = 0.0001$):

$$r^*_A = 1.4276, \qquad l^*_{T^H}(P_A) = 0.06925, \qquad I_{S_A} = 0.001151$$

Note the value of I_{S_A} (integral over the tolerance interval), we will return to it later. Let us add a new point p_{A1} to our set:

$$P_{A1} = P_A \cup \{(0.5, 0.5, 0.4)\}$$

The new values after optimization will be:

$$r^*_{A1} = 0.8557, \qquad l^*_{T^H}(P_{A1}) = 0.09375$$

	r^*	$l^*_{T^H}(P)$	$l_{T^H}(p_{A1})$
$P = P_A$	1.4276	0.06925	0.03033
$P = P_A \cup \{p_{A1}\}$	0.8557	0.09375	0.01978

Table 4.1. Overview of case A, adding point in the center

Before adding the last point, the t-norm would have the value $T^H_{r^*_A}(0.5, 0.5) = 0.2259$, so the loss for this point would be $l = 0.17414^2 = 0.03033$. After optimization, the t-norm tries to compensate for the loss caused by this point by flexing itself in the point's direction. The new loss of the new point for optimized the t-norm will be $l = 0.01978$. So the approximation brought down the loss to 65.23 % of its original value for that point.

What happens when, instead of adding a point in the center, we add a point that is closer to the edge? Let us add a point to our original set P_A that would have the same loss as the point p_{A1} in P_{A1}, but instead lies at $x = 0.9$, $y = 0.5$. For that, we calculate

the value of the t-norm with these arguments and add the distance of p_{A1} from the t-norm before optimization. By doing that, we get a new input data set:

$$P_{A2} = P_A \cup \{(0.9, 0.5, 0.6143)\}$$

Now after optimization we obtain:

$$r^*_{A2} = 1.1477, \qquad l^*_{A2} = 0.09846$$

	r^*	$l^*_{T^H}(P)$	$l_{T^H}(p_{A2})$
$P = P_A$	1.4276	0.06925	0.03033
$P = P_A \cup \{p_{A2}\}$	1.1477	0.09846	0.02094

Table 4.2. Overview of case A, adding point closer to the edge

The new point now has its own loss equal to $l = 0.02094$, which is 69.04 % of its original value, that was the same as for point p_{A1}.

We can see that the data in the center got approximated better – the resulting loss is lower for a point added close to the center than the one added closer to the edge.

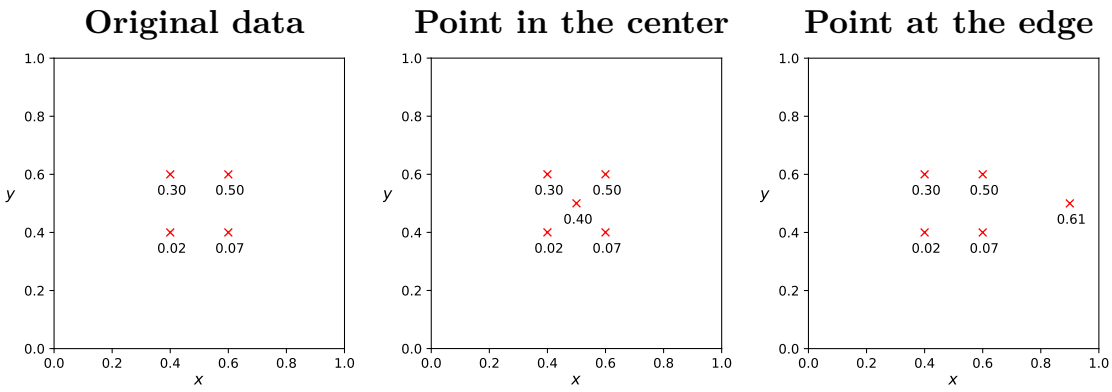

Figure 4.6. Initial data around the center

4.4.4 Example – case B

Now let us consider case B, that is, input data that are closer to *the edge of the unit square*. To be able to compare the results with the previous case, we choose the input data in such a way that the optimal parameter is equal to the one in the previous case:

$$P_B = \{(0.2, 0.2, 0.02), (0.8, 0.2, 0.07), (0.2, 0.8, 0.3), (0.8, 0.8, 0.5726)\}$$

When we approximate these points, we get:

$$r^*_B = 1.4276, \qquad l^*_B = 0.03227, \qquad I_{S_B} = 0.002006$$

The optimal parameter r^*_B is equal to r^*_A, as was intended. Beside that, we can immediately notice that the value of the tolerance integral I_B is almost twice the value of I_A. That means that for approximating data, the t-norm will be able to change its shape more profoundly while maintaining the same precision of the approximation.

Now add a point p_{B1}, which lies in the middle of the unit square:

$$P_{B1} = P_B \cup \{(0.5, 0.5, 0.4)\}$$

	r^*	$l^*_{T^H}(P)$	$l_{T^H}(p_{B1})$
$P = P_B$	1.4276	0.03227	0.030330
$P = P_B \cup \{p_{B1}\}$	0.2123	0.04578	0.007866

Table 4.3. Overview of case B, adding point in the center

Before adding p_{B1}, it would have a loss equal to $l = 0.03033$. After optimizing for the new set P_{B1}, we obtain:

$$r^*_{B1} = 0.2123, \qquad l^*_{B1} = 0.04578$$

With the calculated parameter r^*_{B1}, the point p_{B1} has a loss of 0.007866, which is 25.94 % of the original value.

Now instead of adding a point in the center, we add a point close to the edge of the unit square in the same way as we did in the previous case. That is, we add a point p_{B2} which is equal to p_{A2} because it has the same distance from the t-norm as the points p_{A1} and p_{B1}:

$$P_{B2} = P_B \cup \{(0.9, 0.5, 0.6147)\}$$

For this new set we get:

$$r^*_{B2} = 0.4997, \qquad l^*_{B2} = 0.05814$$

	r^*	$l^*_{T^H}(P)$	$l_{T^H}(p_{B2})$
$P = P_B$	1.4276	0.03227	0.03033
$P = P_B \cup \{p_{B2}\}$	0.4997	0.05814	0.01234

Table 4.4. Overview of case B, adding point to the center

With this optimal parameter r^*_{B2}, the point p_{B2} has a loss of 0.01234. That is 40.69 % of the original value.

Again, the point p_{B1} was approximated better than p_{B2}. This time the difference is even more significant. The reason behind that is that the initial data were clustered around the center in the case A, while in case B they were closer to the edge of the unit square.

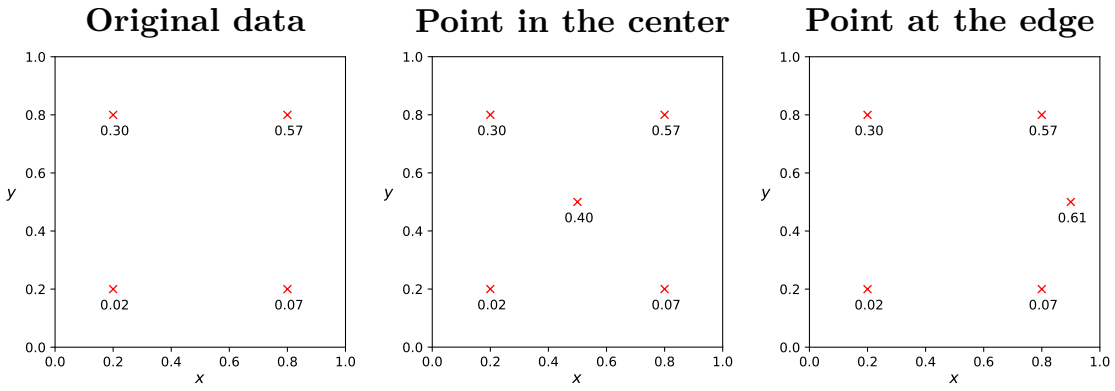

Figure 4.7. Initial data close to the edge

What conclusions can be drawn from this experiment? Chiefly, the closer a data point is to the center of the unit square, the higher influence it has over the optimal

value of parameter r. But that is not a general rule. For that we need to get a deeper insight of exactly how t-norms behave when shifting the value of their parameter. That is of course different for each of the discussed t-norms and for different values of r. This insight can be found in the derivative of the t-norm with respect to the parameter.

4.4.5 Derivative of a t-norm w.r.t. parameter r

To better understand what happens to a t-norm when changing its parameter, let us calculate the derivative:

$$\frac{\mathrm{d}T_r(x,y)}{\mathrm{d}r}$$

When calculating the derivative of the Hamacher t-norm for $r = 1.4276$ (which is the same value on which our previous experiments were based), we get the following:

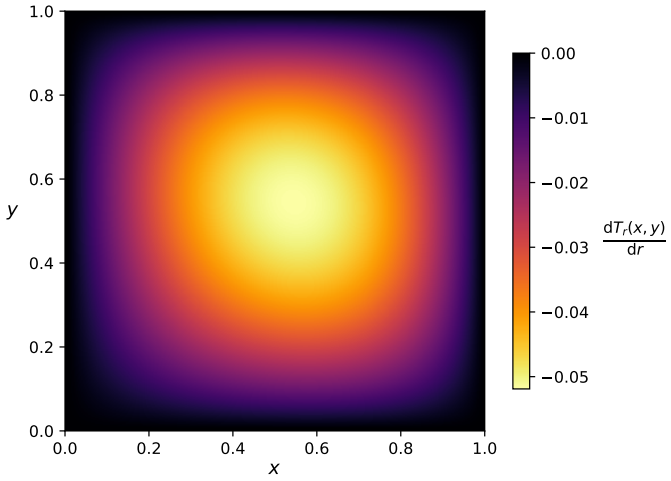

Figure 4.8. Derivative of $T^{\mathrm{H}}_{1.4276}$

What the Figure 4.8 shows us is that the rate of change of the t-norm value, with respect to the parameter r, is higher in the area around the center of the unit square; whereas, closer to the edge, the rate of change is low. This corresponds to our experiments in the previous chapter, where we have shown that values around the center have greater influence over the optimal value of the parameter. Naturally, approximating values in the area with greater values of derivative w.r.t. r will also lead to greater values of the integral I_S.

4.4.6 Non-unimodality of the loss function for the Yager t-norm

An interesting behavior can be observed with the Yager t-norm due to its nilpotency. Let us show an example of the derivative of the Yager t-norm with $r = 0.5$.

As can be seen in Figures 4.9 and 4.10, not only does it have an area on which the value of the t-norm is equal to zero, in the interior of the area the derivative of the t-norm w.r.t. r is also equal to zero. The opposite is true for the edge of the area, where the value of the derivative is the highest.

When we change the parameter r to 1.5, we get the following:

Here we can see that the edge of the area where the t-norm value is zero has moved towards the origin, so has the edge of the area with zero derivative. This moving edge plays an important role during the optimization, because as it moves across data that need to be approximated, the loss function l ceases to be unimodal. Thus, we need to

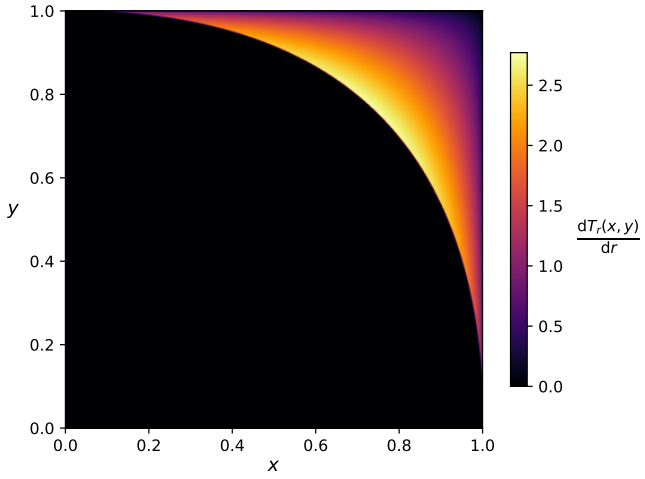

Figure 4.9. Derivative of $T^{\mathrm{Y}}_{0.5}$

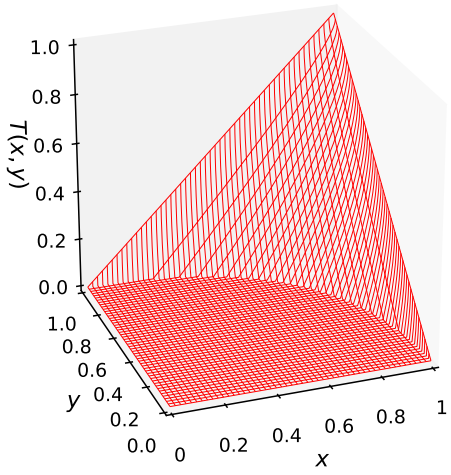

Figure 4.10. Surface of $T^{\mathrm{Y}}_{0.5}$

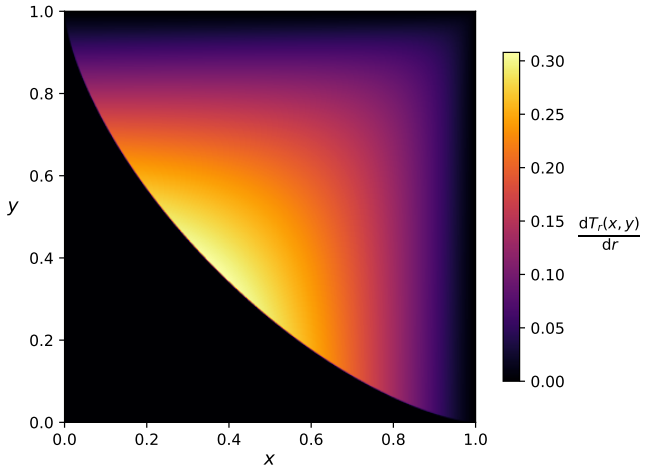

Figure 4.11. Derivative of $T^{\mathrm{Y}}_{1.5}$

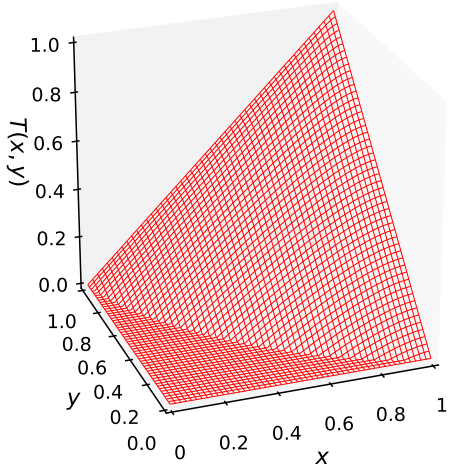

Figure 4.12. Surface of $T^{\mathrm{Y}}_{1.5}$

optimize such t-norms using other techniques – population methods perhaps, such as genetic algorithms or swarm optimization algorithms, as we have done in Section 4.3.

Let us illustrate this behavior through an example. Suppose we need to approximate the following set of points:

$$P = \{p_1, p_2, p_3\},$$

where $p_1 = (0.2, 0.2, 0.1)$, $p_2 = (0.5, 0.5, 0.2)$ and $p_3 = (0.8, 0.8, 0.5)$. For simplicity, we are only considering points lying on the diagonal of the unit square. As was already stated, $T^{\mathrm{Y}}_r \longrightarrow T_D$ for $r \longrightarrow 0$. That means that for large values of the parameter r, the loss function l_r of Yager t-norm for any set of points is given by the sum of squares of their z-coordinates. Let us refer to this value as l_0:

$$l_0 = 0.1^2 + 0.2^2 + 0.5^2 = 0.30$$

When we set r to have the largest value for which $l_r = l_0$, one of the points will suddenly have an influence over the loss function. This is the moment when the edge of the area where $T^{\mathrm{Y}}_r \neq 0$ and $\mathrm{d}T^{\mathrm{Y}}_r(x, y)\,/\mathrm{d}r \neq 0$ reaches the point $p_3 = (0.8, 0.8, 0.5)$, see Fig. 4.13. Until now the loss function has been constant. If we now continue to increase the value of r, the surface of the t-norm will be getting closer to the point p_3

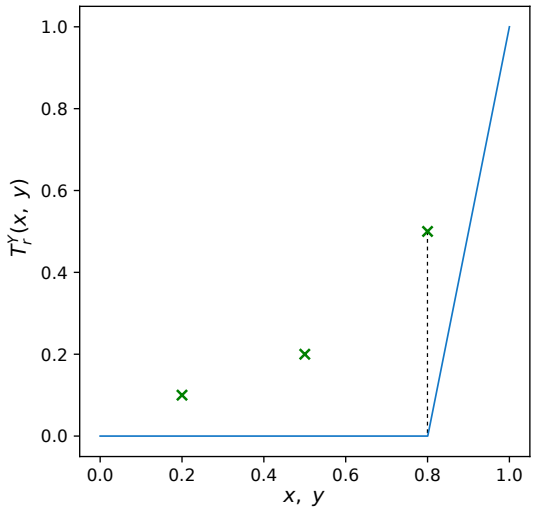

Figure 4.13. Diagonal of $T^{\mathrm{Y}}_{0.43}$

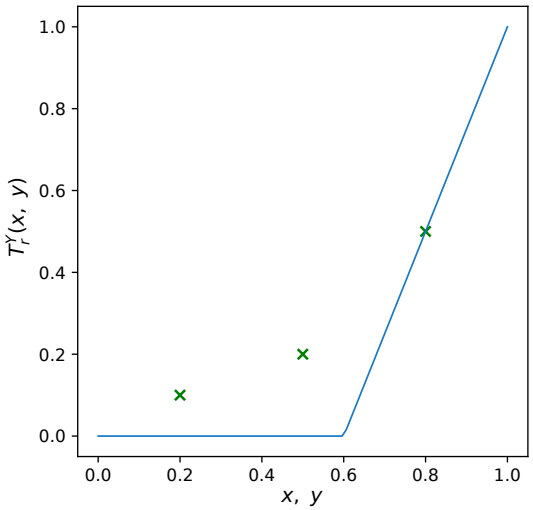

Figure 4.14. Diagonal of $T^{\mathrm{Y}}_{0.76}$

and the value of l_r will be decreasing until the moment when the surface coincides with the point, as can be seen in Fig. 4.14.

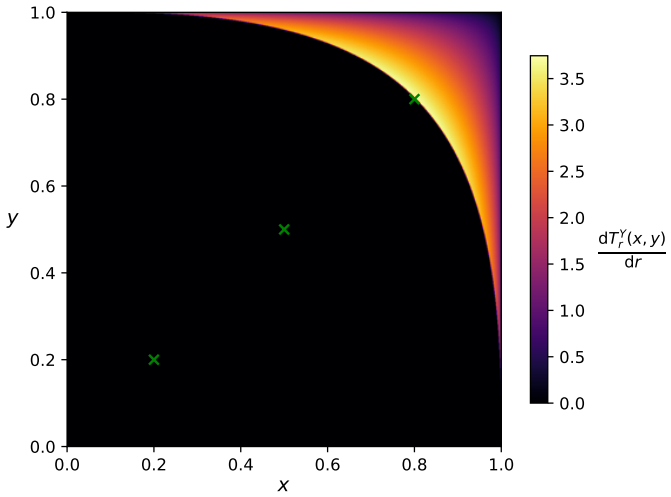

Figure 4.15. Derivative of the $T^{\mathrm{Y}}_{0.43}$

Further increasing the value of r makes the loss function increase because at this moment only the point p_3 has influence on it. That is, the point lies in the area where $\mathrm{d}T_r^{\mathrm{Y}}(x,y)/\mathrm{d}r \neq 0$, but the other points do not. This means that the value of r where the t-norm coincides with p_3 must be a local minimum. The value of the loss function continues to increase until another point p_2 lies in the area of non-zero derivative, see Fig. 4.16. Then both p_2 and p_3 influence the loss function. Now, as can be seen in Fig. 4.18, the rate at which the t-norm is approaching p_2 is greater than the rate at which it recedes from p_3, that is if $p_2 = (x_2, y_2, z_2)$ and $p_3 = (x_3, y_3, z_3)$, we get:

$$\mathrm{d}T_r^{\mathrm{Y}}(x_2, y_2)/\mathrm{d}r > \mathrm{d}T_r^{\mathrm{Y}}(x_3, y_3)/\mathrm{d}r$$

This at first leads to a decrease of the value of the loss function. However, given the fact that squaring in the loss function definition emphasizes larger differences, the distance from the point p_1 becomes too great and the decrease of the loss function stops. Further increase of the value of r would lead to an increase of the value of the loss function. The value of the parameter r for which this is true is $r = 1.27$. As it turns out, this

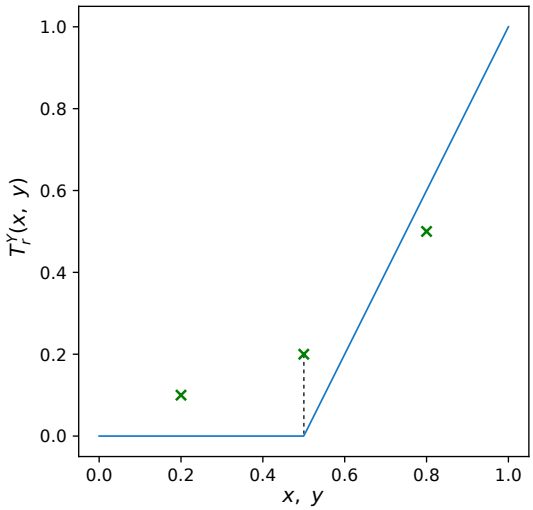

Figure 4.16. Diagonal of $T_{1.0}^{\mathrm{Y}}$

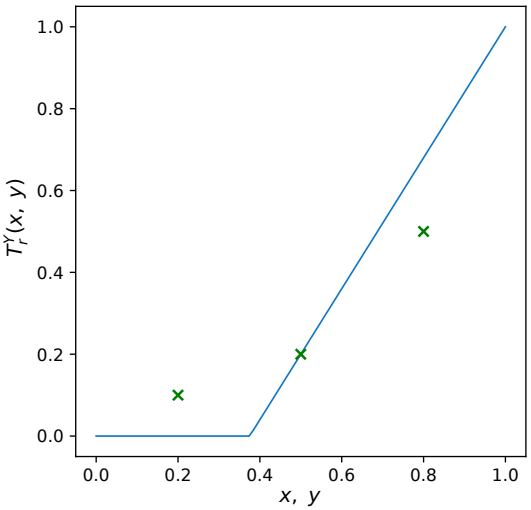

Figure 4.17. Diagonal of $T_{1.47}^{\mathrm{Y}}$

is actually the global minimum we were looking for. The diagonal of the optimized t-norm can be seen in Fig. 4.19.

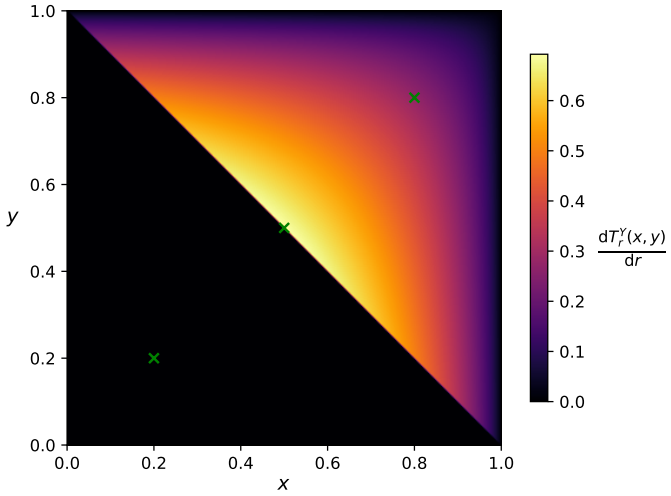

Figure 4.18. Derivative of $T_{1.0}^{\mathrm{Y}}$

If we increased the value of r even more, the loss function would only increase from now on, as can be seen in Fig. 4.20 where both minima are visible.

Importantly, the point p_1 has no influence over the final approximation whatsoever. The optimized parameter r^* would be the same if the initial set of points P did not contain the point. This can be seen from Fig. 4.21. The derivative of the t-norm w.r.t. r for the optimal value r^* is equal to zero at (x_1, y_1), where x_1 and y_1 are the coordinates of the point $p_1 = (x_1, y_1, z_1)$.

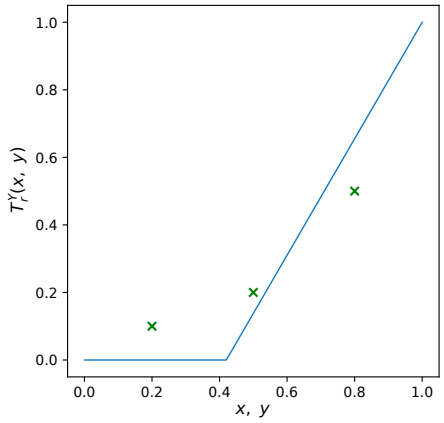

Figure 4.19. Diagonal of $T_{1.27}^{\mathrm{Y}}$

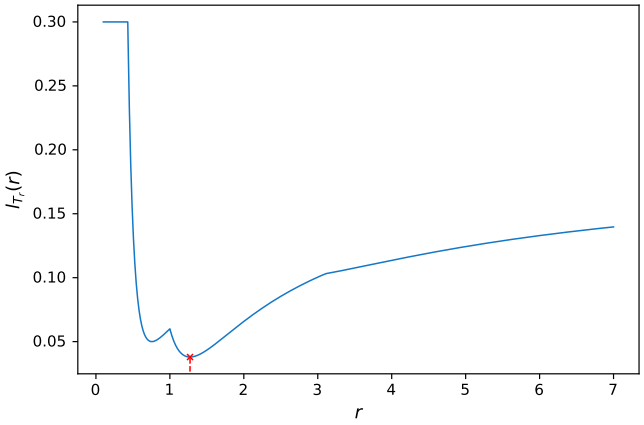

Figure 4.20. Loss function for the given point set

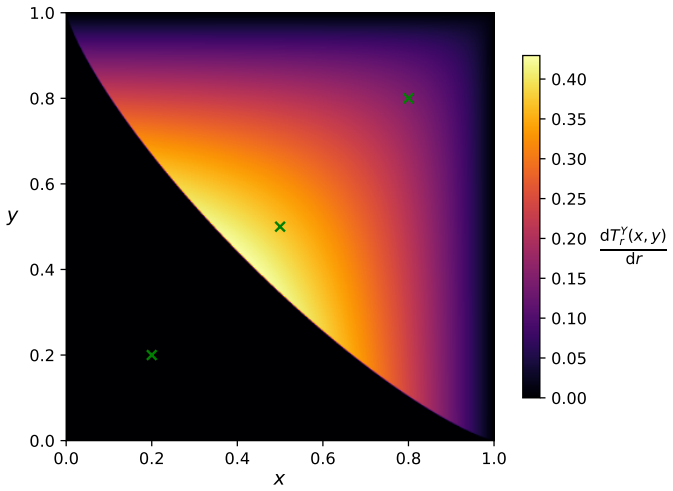

Figure 4.21. Derivative of $T_{1.27}^{\mathrm{Y}}$

The behavior discussed here is the reason why it is problematic to use the Ternary Search algorithm or other simple iterative algorithms to find the global minimum of the loss function. These algorithms are supposed to work well with unimodal functions only, but as we have seen here, the loss function for the Yager t-norm may not be unimodal.

Chapter 5
Optimization of generators

5.1 Generators of t-norms

Additive and multiplicative generators of triangular norms are another way of defining them. Instead of explicitly giving the t-norm's value for a given argument pair (x, y), the value is calculated using a single one-dimensional function, called an additive or a multiplicative generator, and its inverse. In general, we can use generators to produce t-norms which are not continuous. However, that is beyond the scope of this thesis. Here we will deal only with generators of continuos Archimedean t-norms.

5.1.1 Definition

The following definitions are given in [11]:

An *additive generator* of a continuous Archimedean t-norm T is a decreasing bijection $t : [0, 1] \to [0, B], B \in]0, \infty]$, such that:

$$T(x, y) = \begin{cases} t^{-1}(t(x) + t(y)) & \text{if } t(x) + t(y) \leq B, \\ 0 & \text{otherwise.} \end{cases}$$

We can also define n-nary extension of a t-norm given by its additive generator [7]. If $t : [0, 1] \to [0, B], B \in]0, \infty]$ is an additive generator of a t-norm T, we have for all $x_1, x_2, \cdots, x_n \in [0, 1]$

$$T(x_1, x_2, \cdots, x_n) = \begin{cases} t^{-1}\left(\sum_{i=1}^{n} t(x_i)\right) & \text{if } \sum_{i=1}^{n} t(x_i) \leq B, \\ 0 & \text{otherwise.} \end{cases}$$

A *multiplicative generator* of a continuous Archimedean t-norm T is an increasing bijection $\theta : [0, 1] \mapsto [b, 1], b \in [0, 1[$, such that:

$$T(x, y) = \begin{cases} \theta^{-1}(\theta(x) \cdot \theta(y)) & \text{if } \theta(x) \cdot \theta(y) \geq b, \\ 0 & \text{otherwise.} \end{cases}$$

Again, we can define an n-ary extension. If $\theta : [0, 1] \mapsto [0, \infty]$ is a multiplicative generator of a t-norm T, we have for all $x_1, x_2, \cdots, x_n \in [0, 1]$ [7]:

$$T(x_1, x_2, \cdots, x_n) = \begin{cases} \theta^{(-1)}\left(\prod_{i=1}^{n} \theta(x_i)\right) & \text{if } \prod_{i=1}^{n} \theta(x_i) \geq b, \\ 0 & \text{otherwise.} \end{cases}$$

Generators of t-norms are not uniquely determined by their t-norm. That is, a single generator function results in a unique triangular norm, but the same t-norm might be generated using other generator functions as well.

5.1.2 Examples

Examples of additive and multiplicative generators and their respective t-norms can be found in [5]. Here we show just some of them:

- An additive generator of the Łukasiewicz t-norm T_{L} is given by:

$$t(x) = 1 - x,$$

  Note that this leads to:

$$t^{-1}(t(x) + t(y)) = 1 - ((1 - x) + (1 - y)) = 1 - (2 - x - y),$$

  which is a function not bounded by $[0, 1]$ for $x, y \in [0, 1]$. In this case, we would need to additionally limit the resulting function to the proper bounds.
- An additive generator of the product t-norm T_{P}:

$$t(x) = -\log x$$

- An additive generator of the Frank t-norm T_r^{F} with parameter r:

$$t_r^{\mathrm{F}}(x) = \begin{cases} -\log x & \text{if } r = 1, \\ 1 - x & \text{if } r = \infty, \\ \log \frac{r-1}{r^x - 1} & \text{if } r \in \,]0, 1[\,\cup\,]1, \infty[. \end{cases}$$

- A multiplicative generator of the Frank t-norm T_r^{F} with parameter r:

$$\theta_r^{\mathrm{F}}(x) = \begin{cases} x & \text{if } r = 1, \\ \mathrm{e}^{x-1} & \text{if } r = \infty, \\ \frac{r^x - 1}{r-1} & \text{if } r \in \,]0, 1[\,\cup\,]1, \infty[. \end{cases}$$

- An additive generator of the Hamacher t-norm T_r^{H} with parameter r:

$$t_r^{\mathrm{H}}(x) = \begin{cases} \frac{1-x}{x} & \text{if } r = 0, \\ \log\left(\frac{r+(1-r)x}{x}\right) & \text{if } r \in \,]0, \infty[. \end{cases}$$

- A multiplicative generator of the Hamacher t-norm T_r^{H} with parameter r:

$$\theta_r^{\mathrm{H}}(x) = \begin{cases} \mathrm{e}^{\frac{x-1}{x}} & \text{if } r = 0, \\ \frac{x}{r+(1-r)x} & \text{if } r \in \,]0, \infty[. \end{cases}$$

Generators of the parametric t-norms for several values of the parameter r can be seen in the figures below.

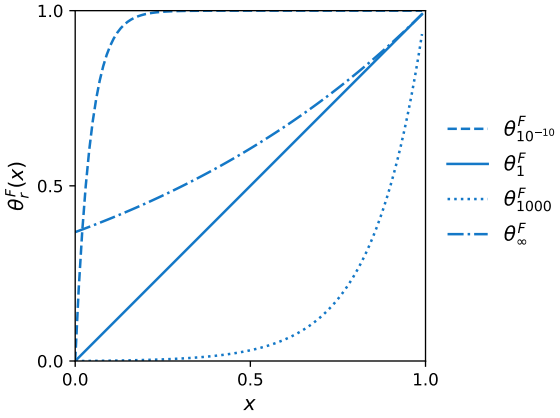

Figure 5.1. Multiplicative generators of T_r^{F}

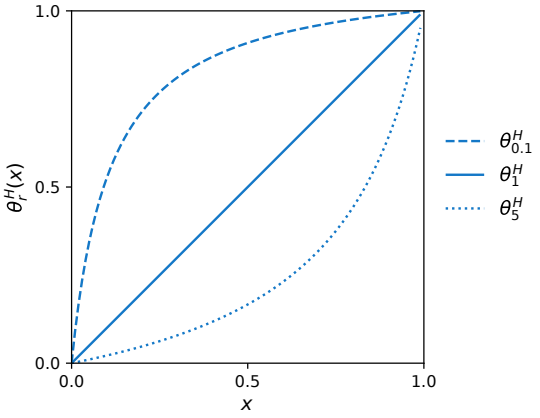

Figure 5.2. Multiplicative generators of T_r^{H}

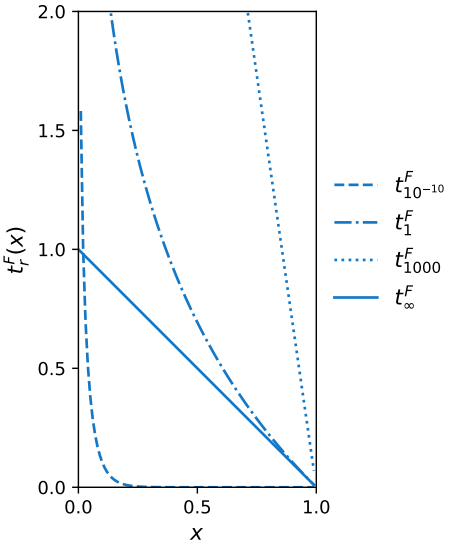

Figure 5.3. Additive generators of T_r^{F}

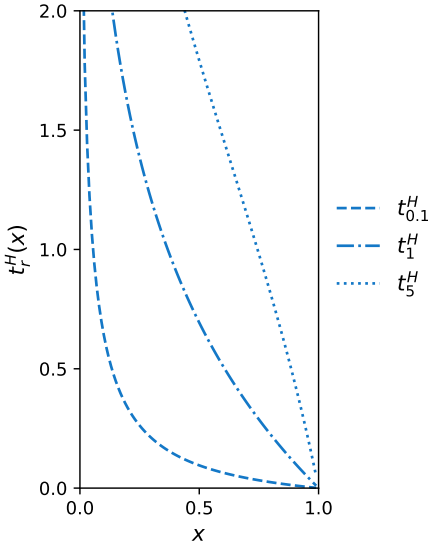

Figure 5.4. Additive generators of T_r^{H}

5.2 Fitting generators of general t-norms

As is stated in Proposition 2.12.7 of [7], the convergence property of a sequence of t-norms $\lim_{n\to\infty} T_n = T$ implies the existence of a sequence of additive generators $(t_n : [0,1] \mapsto [0,\infty])_{n\in\mathbb{N}}$ of T_n. The restriction of $(\lim_{n\to\infty} t_n)|_{]0,1]}$ coincides with a restriction of some additive generator of T to $]0,1]$. For strict t-norms, these restrictions are $[0,1]$. This fact allows us to deal with approximating additive generators instead of their t-norms as we can transform one problem to the other.

For simplicity, let us consider the case of approximation by a strict t-norm; the conclusions will be relevant to nilpotent t-norms, too.

Let us denote by t an additive generator of the t-norm we are approximating. As was already stated, for an input dataset consisting of tuples $(x_i, x_j, y_{i,j}), \quad i,j \in \{1,\dots,n\}$, we want to find an approximate solution to the system of equations:

$$t^{-1}(t(x_i) + t(x_j)) \approx y_{i,j}\,, \qquad i,j \in \{1,\dots,n\}\,. \tag{1}$$

for an additive generator t, or

$$\theta^{-1}(\theta(x_i) \cdot \theta(x_j)) \approx y_{i,j}\,, \qquad i,j \in \{1,\dots,n\}\,. \tag{2}$$

for a multiplicative generator θ. We will focus on the additive generator approach for now. We can choose to optimize this system of equations directly, that is, try to find an approximate solution to this system by minimizing:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\big(t^{-1}(t(x_i) + t(x_j)) - y_{i,j}\big)^2, \tag{3}$$

or we can modify the system in a way that could somehow help us to find the solution more easily. We will refer to the unmodified criterion (3) as a *proper criterion*. Beliakov [1] proposes a modification of (1) to avoid calculating the inverse of the function g by instead finding an approximate solution to the following system of equations:

$$t(x_i) + t(x_j) \approx t(y_{i,j})\,, \qquad i,j \in \{1,\dots,n\}\,. \tag{4}$$

Not all of the values $y_{i,j}$, $i, j \in \{1, \ldots, n\}$ need to be given. This modification allows us to also formulate the problem as an LSEI problem (least squares with equality and inequality constraints). We will not go into detail about how exactly this is achieved. For now, let us just note that a number of solvers of this problem exist. During experiments in this work, an evolution-based algorithm was used to minimize the loss functions. This naturally results in a significant decrease in performance. Nevertheless, it allows us to easily test different criteria without the need to manually transform the equations from one formulation to another. This results in being able to use the same solvers, which is needed to properly compare the results. Given the tight time frame, this would not be feasible.

In [1], also prescribed values of the t-norm for more than two arguments are allowed; this generalization is obtained almost for free. We do not consider this case here for simplicity. Besides, our method of generating data was not designed for this general case.

The approximate solution to system (4) can be found by the minimization of a criterion, e.g. the least squares,

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\big(t(x_i) + t(x_j) - t(y_{i,j})\big)^2 . \tag{5}$$

Note, that we will refer to this criterion as the *original criterion*. As the data are finite, the criterion depends only on the values of g at finitely many points,

$$x_1, x_2, \ldots, x_n, y_{1,1}, y_{1,2}, \ldots, y_{n,n} . \tag{6}$$

Between them, any continuous monotonic function can be admitted. Beliakov [1] uses piecewise linear interpolation and the reciprocal function ($x \mapsto 1/x$) in the interval between 0 and the least value, $y_{1,1}$.[1]

Criterion (5) allows us to avoid the use of the inverse of the generator as mentioned above. However, the criterion has a trivial zero solution of $t(x) = 0$. Thus we need a "fixed unit" that prevents us from considering this trivial solution. In [1], this is taken as $t(0) = 1$ in the case of a nilpotent t-norm and $t(\varepsilon) = 1$ in the case of a strict t-norm, where ε is a properly chosen small positive number. The author takes the smallest value from points (6). On the interval $[0, \varepsilon]$, the author proposes modelling the asymptote near zero using the function $1/x$. He also suggests that the choice of the function in this interval has no influence on the final approximation. We do not agree with this statement and will provide the explanation later.

The author of [1] considers the choices of ε, spline knots and the the interpolation functions between the knots to be satisfactory. We dare to formulate some criticism.

The choice of the generator and the optimality criterion can have problematic consequences. Recall that an additive generator $t(x)$ is a monotonically decreasing function. If we pick ε as the minimum of our input data (e.g. $\varepsilon = y_{1,1}$), all other input values will lead to lower values of t than $t(\varepsilon) = 1$. Because smaller values of t lead to smaller values of criterion (5), we may expect that the "optimized" value of $t(y_{1,1})$ would be rather "high", relative to other values of t. That is, it would have higher influence over the generator than the other input data points. One may expect that the estimate of

[1] In [1], the author also suggests that the number and choice of the nodes of linear splines used for approximation can be modified to control the precision of the resulting approximation and thus they can be independent of the approximated points. He also suggests splines of a higher order as an alternative, provided that their monotonicity is ensured, which is trivially satisfied for the linear spline.

$t(y_{1,1})$ would be "biased" (not in the statistical sense). The same goes for all of the input values. The lower the input value is, the higher influence over the generator it has. This presents a larger problem when the number of knots of the B-spline used to approximate the generator is lower. This, of course, results in lower precision of the final approximation.

Therefore, it is highly desirable to use an optimality criterion that refers to the t-norm, not to its generator, which not only skews the weights of contributions of different data points to the value of the optimality criterion, it is also not unique.

In order to evaluate the difference of the values of the t-norm, we need to reconstruct its values; equations (4) do not refer to them. To be able to apply this idea exactly, we would need to evaluate the differences

$$t^{-1}\big(t(x_i) + t(x_j)\big) - y_{i,j}\,, \qquad i,j \in \{1,\ldots,n\}\,.$$

These require the values of the generator at points different from those of (6) as here we also need to have the inverse of the generator t^{-1}. Then the method of interpolation between these points becomes important. Not only that, the choice of the function modelling the asymptote near zero is also important because the sum $t(x_i) + t(x_j)$ might be larger than 1. This means that the inverse of this sum would fall into the interval $[0, \varepsilon]$.

This sums up the reasons of why the choice of interpolating functions between the b-spline knots and the function on the interval $[0, \varepsilon]$ in fact do play a significant role in the resulting approximation quality. However, due to computational overhead, it might be highly problematic to use a criterion with the inverse of the generator during the optimization phase. For that, a criterion similar to (5) might be sufficient. Nevertheless, that criterion results in a t-norm that is "biased", as has been explained earlier.

To solve this, we can suggest an approximate criterion for optimization based on the relative error of the estimate of the solution to (4) and the desired value $y_{i,j}$, e.g.

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\left(\frac{t(x_i)+t(x_j)}{t(y_{i,j})} - 1\right)\cdot y_{i,j}\right)^2\,. \tag{7}$$

This criterion compensates for the bias introduced by the non-linearity of the generator and provides better results than (5). To demonstrate this, we randomly generated 100 datasets of 50 points using the product t-norm T_{P}. Then we optimized a generator consisting of the function $1/x$ in the interval $[0, \varepsilon]$ and a spline with 5 knots placed equidistantly between 1 and ε. The value of ε was chosen independently for each dataset as the smallest value from the input data points. An example of such a generator can be seen in Figure 5.5. Note that the number of knots is not the same as a number of parameters of the optimization. The first knot at ε and the last knot at 1 are fixed with the first being dependant on the input data. Only the rest of the knots are then being optimized.

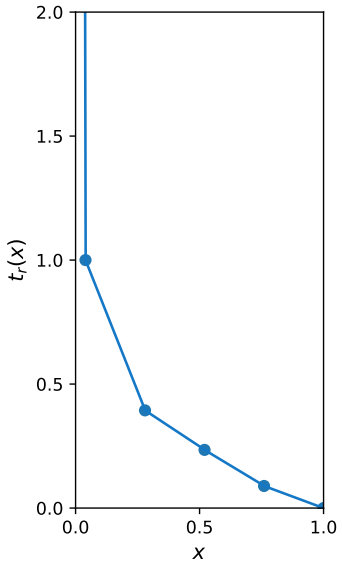

Figure 5.5. Example of a generator with 5 equidistant knots

Upon optimizing for the 100 datasets, we obtained the results that are visualized in Figure 5.6. There, the *predicted* values are those values $y_{i,j}$ that we obtain from the generator t optimized for the given dataset, that is, $y_{i,j} = t^{-1}(t(x_i)+t(x_j))$. The *actual* values correspond to the reference values in the dataset, that is, the values of the t-norm T that we are trying to approximate. [1] The closer a point is to the diagonal (visualized as a black line), the more precise the resulting approximation was. From the figure we can notice that both methods struggle with approximating values close to zero. Thus, they might not be suitable for approximating nilpotent t-norms.

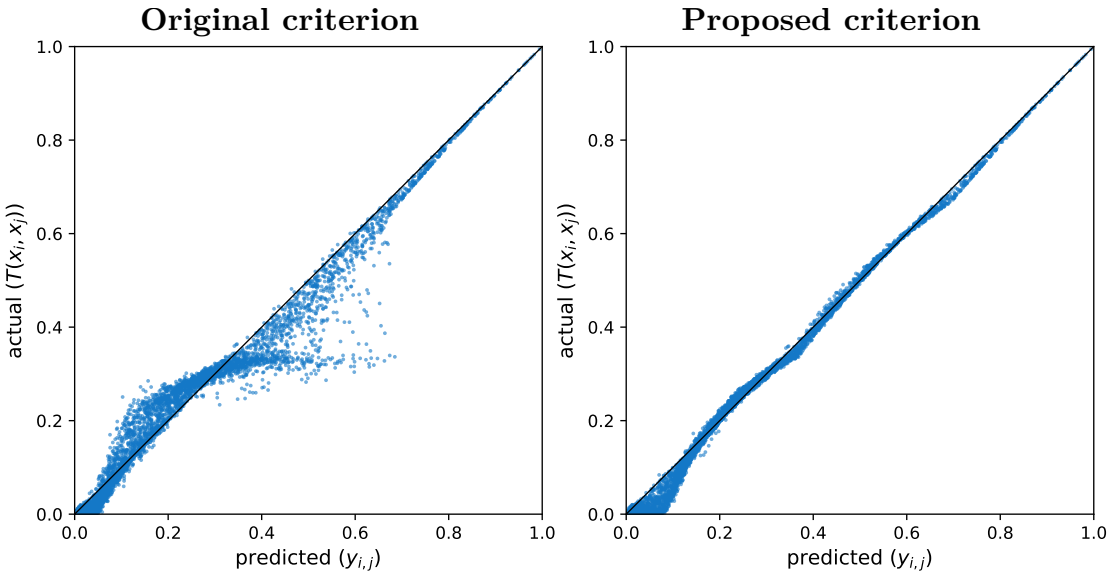

Figure 5.6. 50 pts. × 100 iterations, 5 equidistant knots

[1] Here, of course, the t-norm is known, because we used it to generate the training data.

To compare the approximation precision, we use the root mean square error (RMSE). In our case, this is defined as:

$$\mathrm{RMSE} = \sqrt{\left(\sum_{(x_i,x_j,y_{i,j})\in P} \left(t^{-1}(t(x_i)+t(x_j)) - y_{i,j}\right)^2\right) / |P|},$$

where P is the input dataset and $|P|$ is the number of points in the dataset.

For datasets consisting of 50 points and the number of knots in the linear b-spline forming the generator being 5, the average RMSE of each dataset was equal to 0.0438 when using criterion (5) and 0.0233 when using the criterion (7) with the same input data. We can see a significant improvement of the approximation quality.

The same is true even for datasets of different sizes and different numbers of knots. For larger datasets of 100 points, the RMSE was 0.0461 for the original criterion and 0.0175 for the proposed criterion. In Figure 5.7, we used smaller datasets of 15 points and also increased the number of knots in the splines to 9. The resulting RMSE for the original and the proposed criterion were 0.0134 and 0.0042, respectively. Again, we can see an improvement in the approximation precision.

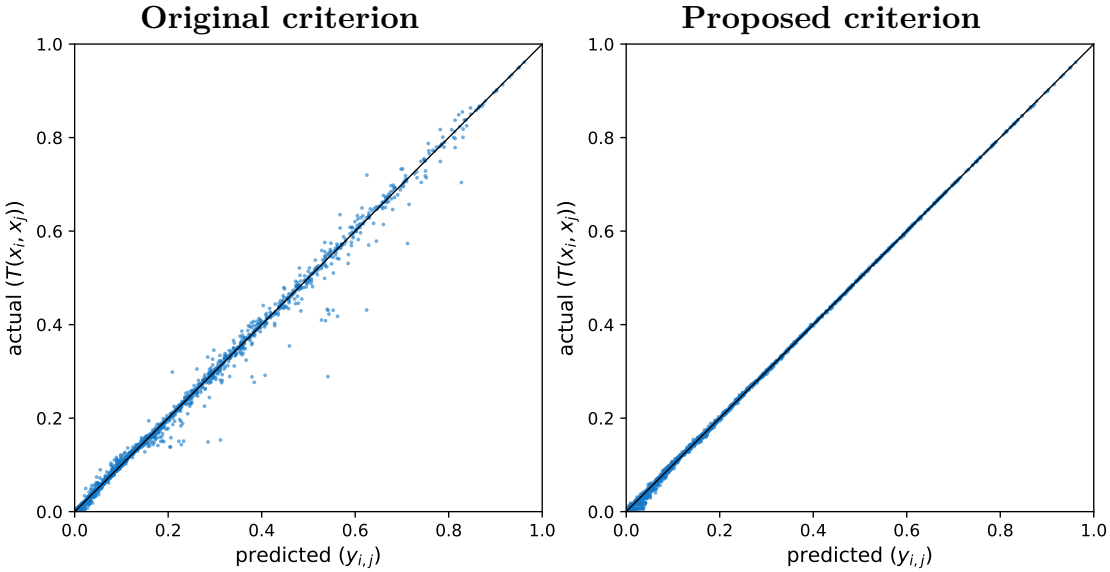

Figure 5.7. 15 pts. × 100 iterations, 9 equidistant knots

More selected predicted vs. observed plots can be found in Appendix C.

In Table 5.1 we provide results of some of the experiments that were made using datasets of different size and generated by different t-norms. Also, the number of knots k was altered. In the table, we can see the values of the original criterion (5), the proposed criterion (7) as well as the proper criterion (3). Unsurprisingly, it is clear that the proper criterion outperforms the other criteria. However, the proposed criterion has a better precision of approximation than the original criterion in most cases.

As seen above, the proposed criterion can improve the approximation quality significantly. Additionally, it can be optimized using the same techniques as those suggested in [1]. However, the proposed criterion is performing noticeably worse when approximating data containing a larger number of zeros, such as the case of the Yager t-norm with $r = 0.3$. There, the original criterion outperforms the one we propose.

Examples of generators that were results of some of the experiments can be seen in Figures 5.8, 5.9 and 5.10. Be aware that those are illustrative plots serving as an

ref. t-norm	k	$\lvert P\rvert$	$\mathrm{RMSE}_{\mathrm{orig.}}$	$\mathrm{RMSE}_{\mathrm{proposed}}$	$\mathrm{RMSE}_{\mathrm{proper}}$
T_{P}	5	5	0.0169	0.0089	0.0083
T_{P}	5	50	0.0395	0.0165	0.0149
T_{P}	5	100	0.0388	0.0167	0.0152
T_{P}	7	8	0.0317	0.0096	0.0049
T_{P}	9	15	0.0192	0.0048	0.0044
$T^{\mathrm{F}}_{10^{-5}}$	5	5	0.0795	0.0248	0.0099
$T^{\mathrm{F}}_{10^{-5}}$	5	50	0.1130	0.0182	0.0160
$T^{\mathrm{F}}_{10^{5}}$	5	5	0.0474	0.0072	0.0053
$T^{\mathrm{F}}_{10^{5}}$	5	50	0.0633	0.0096	0.0079
T^{Y}_{2}	5	5	0.0514	0.0162	0.0091
$T^{\mathrm{Y}}_{0.3}$	5	5	0.1375	0.2635	0.0855

Table 5.1. Overview of RMSE for different experiments

example and they do not represent all the generators that were created as a part of certain experiments. In fact, the resulting generators varied significantly for different input datasets, even when the data where created using the same t-norm and were the same in size. More figures of resulting generators can be found in Appendix B.

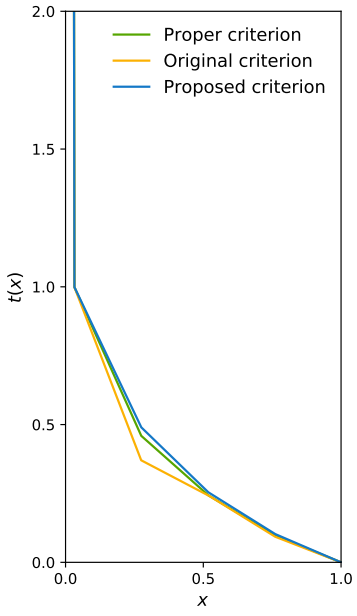

Figure 5.8. $t(x)$ for T_{P}, $|P| = 50$, $k = 5$

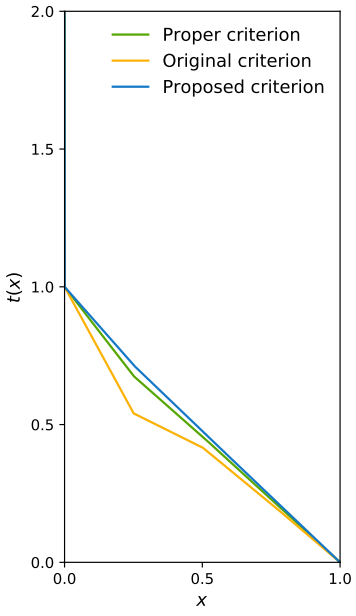

Figure 5.9. $t(x)$ for $T^{\mathrm{F}}_{10^5}$, $|P| = 50$, $k = 5$

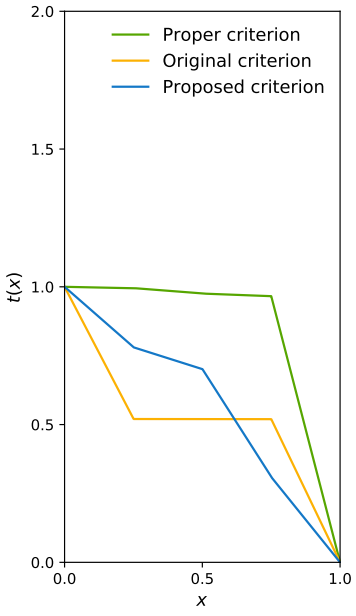

Figure 5.10. $t(x)$ for $T^{\mathrm{Y}}_{0.1}$, $|P| = 50$, $k = 5$

Another fact that we wanted to point out about the approach in [1] is that the same approach with additive generators could also be applied to multiplicative generators.[1] Then the contribution of errors in different points would change; we would reach another optimum and another t-norm. As is stated in [5], we can obtain a multiplicative

[1] Then also the linear or other interpolation may have a different meaning, but we do not discuss it because this cannot be evaluated based only on the given data; additional criteria, e.g., success in a collection of real-life problems, could distinguish different methods of interpolation.

generator $\theta(x)$ from an additive generator $t(x)$ by putting $\theta(x) = \exp(-t(x))$. If the multiplicative generator is a function $x \mapsto \exp(-t(x))$, its optimum would correspond to a weighted form of (5),

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\left(t(x_i) + t(x_j) - t(y_{i,j})\right)^2, \tag{8}$$

with weights $w_{i,j}$, $i, j \in \{1, \ldots, n\}$ proportional to $y_{i,j}$ reflecting the transition from an additive generator to a multiplicative generator.

However, there is a more serious obstacle: there are more multiplicative generators and the criterion of the form (5) leads to different optima, dependent on the choice of the generator.

Returning back to our question regarding what criterion to choose for the optimization, we could in fact choose the most obvious road, that is, optimizing (1) directly by minimizing the proper criterion (3). Suppose we continue using function t in the same form, i.e., a monotonic b-spline of order 1 on the interval $]\varepsilon, 1]$. Also, suppose we use the exponential function on the interval $[0, \varepsilon]$. Then, taking inverse of this function is a straightforward task and optimizing the system would yield even better results than all the criteria discussed here. In fact, it will give us the best approximation precision possible as the value of RMSE is computed using the same differences or error terms.

5.3 Comparison to approximation by parametric t-norms

There is no doubt that having a priori information about the input data, like what kind of triangular norm is expected, could significantly help to improve the quality of our approximation. That is, how precise our predictions will be with the approximated t-norm. This information could also be a decisive factor of whether to approximate by parametric t-norms or by approximating the searched t-norm's additive or perhaps multiplicative generator.

When there is no such information available, using generators provides the most flexibility in terms of being able to fit any data with relatively high precision. On the other hand, it could also make things unnecessarily complicated because we would need to decide what kind of interpolation function to choose between the knots of the generator, the number of knots, which function to choose to model the asymptote near zero when dealing with strict t-norms, and finally, which criterion to choose. The answers to these questions need a more experiments and they also depend on the demands we impose on the speed and precision of the approximation.

Chapter 6
Conclusion

In this thesis we tried to give a basic overview of properties of fuzzy conjunctions (triangular norms) as well as examples of different classes of triangular-norms. Fundamental definitions and notions were introduced and different techniques of approximating data using both parametric t-norms and general t-norms determined by additive generators were explored.

First, we created a simple random data generator producing data that are monotone and can be tweaked to meet our needs. The random data generator can be easily modified to yield data of great variability and can be used to model random distributions of data from different t-norms.

The generated data correspond to a triangular norm which we would like to approximate. As a target of our optimization, we used a loss function equal to the sum of squared errors. Hence, the optimization method was in fact the least squares method. First, we tried to fit them by some of the parametric t-norms, that is, triangular norms that are determined by an additional parameter (besides its arguments). For the optimization itself, two methods were used. The first was the Ternary Search algorithm which was relatively easy to implement and was computationally efficient and therefore fast. It could be improved in terms of optimizing the initial size of the search interval and experimenting with different methods of choosing the cut points of the search interval. This would lead us to modifications of this algorithm, such as the Golden Section Search. In practice this was not necessary because the basic version of the Ternary Search was fast enough for our needs. For big datasets, however, one should consider to implement the optimizations considered above. It was found out, however, that the algorithm was not usable for optimizing the Yager t-norms. The reason was that when using nilpotent t-norms, we get a loss function that is not unimodal and the Ternary Search algorithm can converge to any local minimum instead of the global minimum. Thus we needed to use some other approach to optimize nilpotent t-norms.

A solution we chose was a population-based algorithm, namely the particle swarm optimization algorithm. This algorithm, albeit computationally far more demanding than the Ternary Search algorithm and its other versions, allowed us to find a global optimum of the loss function of any t-norm. Again, the algorithm could be optimized to reach the optimal values faster and we will definitely consider doing that in the future.

To assess the result of the approximation, we used the volume integral of t-norms for certain values of their parameter given by a specific approximation precision interval. This integral allowed us to study how input data distribution influences the precision or "stability" of the resulting approximation. The results were also explained using the derivative of the t-norms with respect to the parameter. Using these tools, we also explained why the loss function of the Yager t-norms and at the same time other nilpotent t-norms is not always unimodal and therefore why it is needed to use other means of optimization than those used for strict t-norms.

Besides parametric t-norms, we also experimented with t-norms determined by their additive generators. We proposed possible improvements to already existing methods

of approximating data by optimizing additive generators. Namely, a new optimization criterion was proposed. This criterion mitigates the original criterion's flaw, which is creating a bias towards data points with lower value. Indeed, the proposed criterion proved to yield better results in most cases, with the exception being input data that have data points with the value of zero or simply very low. But those are data that the original criterion has problems with, too.

More experiments will have to be conducted in order to properly evaluate what kinds of approximation are best suited for certain types of data. Furthermore, it would be highly useful to test means of interpolation between knots in generators of t-norms other than the linear interpolation. The same applies to the function in the interval $[0, \varepsilon]$, which, as we have shown, also has influence over the quality of the final approximation.

Further testing of performance of various algorithms mentioned here will also be needed. Some of the calculations for the experiments were running for a relatively long time and waiting too long for a certain optimization task to finish is not always acceptable. Similarly, the performance of solvers for the LSEI tasks of the generator-based approximation techniques could be tested to see the real performance difference between the various optimization criteria. It is also worth investigating whether or not it is viable to optimize the differences of the input data and generated t-norm directly.

References

[1] Gleb Beliakov. Fitting triangular norms to empirical data. In Erich Peter Klement and Radko Mesiar, editors, *Logical, Algebraic, Analytic and Probabilistic Aspects of Triangular Norms*, pages 261 – 272. Elsevier Science B.V., Amsterdam, 2005.

[2] Jørgen Harmse. Continuous fuzzy conjunctions and disjunctions. *IEEE Transactions on Fuzzy Systems*, 4(3):295–314, 1996.

[3] James Kennedy and Russel Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[4] Jack Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4:502–506, 1953.

[5] Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular Norms*. Springer, Dordrecht, Netherlands, 2000.

[6] Erich Peter Klement, Radko Mesiar, and Endre Pap. Invariant copulas. *Kybernetika*, 38(3):275–286, 2002.

[7] Erich Peter Klement, Radko Mesiar, and Endre Pap. Triangular norms: Basic notions and properties. In Erich Peter Klement and Radko Mesiar, editors, *Logical, Algebraic, Analytic and Probabilistic Aspects of Triangular Norms*, pages 17 – 60. Elsevier Science B.V., Amsterdam, 2005.

[8] Jan Łukasiewicz. *Die Logischen Grundlagen der Wahrscheinlichkeitsrechnung*. Akademie der Wissenschaften, Kraków, 1913. English translation: Jan Lukasiewicz and Ludwik Borkowski. *Selected works by Jan Łukasiewicz*. Amsterdam, pages 16–63, 1970.

[9] Jan Łukasiewicz. O logice trójwartościowej. *Ruch filozoficzny*, 5:170–171, 1920. English translation: Jan Lukasiewicz and Ludwik Borkowski. *Selected works by Jan Łukasiewicz*. Amsterdam, pages 87–88, 1970.

[10] Karl Menger. Statistical metrics. *Proceedings of the National Academy of Sciences, USA*, 28:535–537, 1942.

[11] Mirko Navara. Triangular norms and conorms. *Scholarpedia*, 2(3):2398, 2007. revision #137537.

[12] Roger B. Nelsen. Copulas and quasi-copulas: An introduction to their properties and applications. In Erich Peter Klement and Radko Mesiar, editors, *Logical, Algebraic, Analytic and Probabilistic Aspects of Triangular Norms*, pages 391 – 413. Elsevier Science B.V., Amsterdam, 2005.

[13] Michail I. Schlesinger and Václav Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition*. Kluwer Academic Publishers, 2002.

[14] Berthold Schweizer and Abe Sklar. Espaces métriques aléatoires. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Paris*, 247:2092–2094, 1958.

[15] Berthold Schweizer and Abe Sklar. Statistical metric space, pacific. *J. Math.*, 10:313–334, 1960.

[16] Berthold Schweizer and Abe Sklar. Associative functions and statistical triangle inequalities. *Publicationes Mathematicae Debrecen*, 1961.

[17] Lotfi Aliasker Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

Appendix A
Specification

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Popovič Alexej** Personal ID number: **466324**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Approximation by Fuzzy Conjunctions

Bachelor's thesis title in Czech:

Aproximace hodnot fuzzy konjunkcemi

Guidelines:

Make a survey of construction techniques of fuzzy conjunctions (triangular norms). Analyse the possibilities of fitting a fuzzy conjunction to given data. You may choose from known classes of fuzzy conjunctions or use general fuzzy conjunctions determined by their generators.
The thesis should consist of an overview of theoretical principles, proposed method(s) of approximation, implementation, and experimental verification.

Bibliography / sources:

[1] Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms, vol. 8 of Trends in Logic. Kluwer Academic Publishers, Dordrecht, Netherlands, 2000.
[2] Petrík, M., Sarkoci, P.: Zero-reconstructible triangular norms as universal approximators. Neural Network World 10 (2010), 63-67.
[3] Mesiar, R.: Approximation of continuous t-norms by strict t-norms with smooth generators. BUSEFAL 75 (1998) 72–79.
[4] Navara, M., Petrík, M.: Generators of fuzzy logical operations. In: H.T. Nguyen, V. Kreinovich (eds.), Alge-braic Techniques and Their Use in Describing and Processing Uncertainty, Studies in Computational Intelligence 878, Springer, 2020. DOI 10.1007/978-3-030-38565-1_8

Name and workplace of bachelor's thesis supervisor:

prof. Ing. Mirko Navara, DrSc., Machine Learning, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **07.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

prof. Ing. Mirko Navara, DrSc.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt

Student's signature

Appendix B
Selected generators from experiments

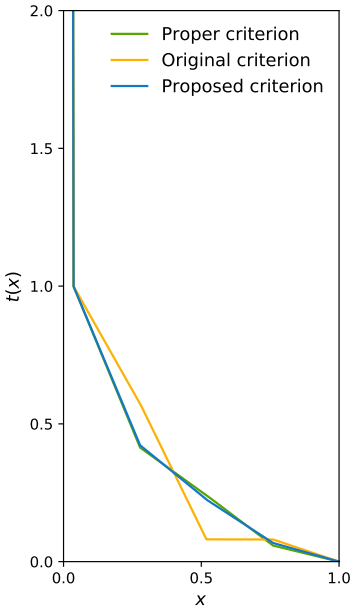

Figure B.1. $t(x)$ for T_{P}, $|P| = 5$, $k = 5$

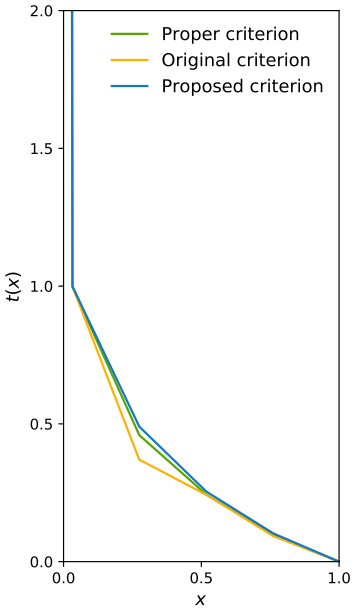

Figure B.2. $t(x)$ for T_{P}, $|P| = 50$, $k = 5$

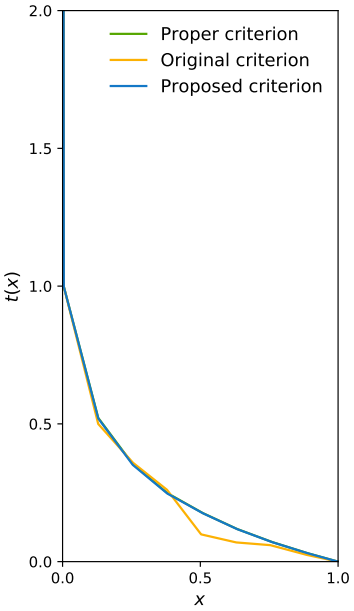

Figure B.3. $t(x)$ for T_{P}, $|P| = 15$, $k = 9$

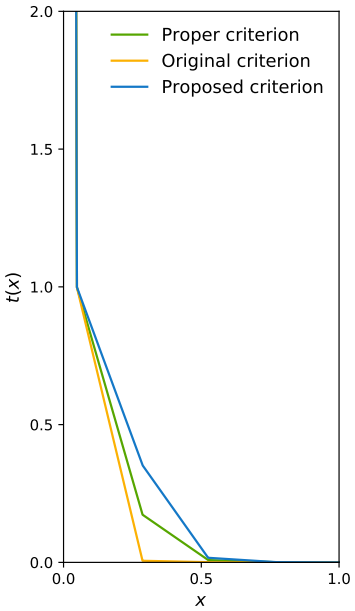

Figure B.4. $t(x)$ for $T^{\mathrm{F}}_{10^{-5}}$, $|P| = 5$, $k = 5$

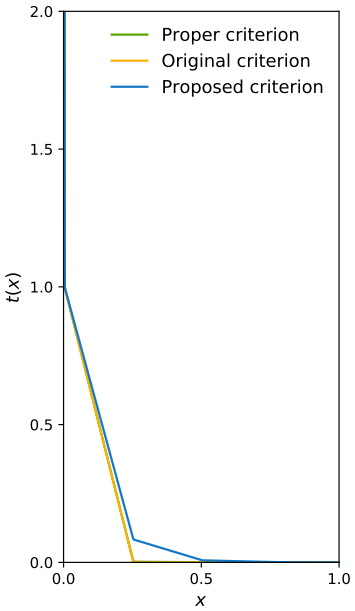

Figure B.5. $t(x)$ for $T^{\mathrm{F}}_{10^{-5}}$, $|P| = 50$, $k = 5$

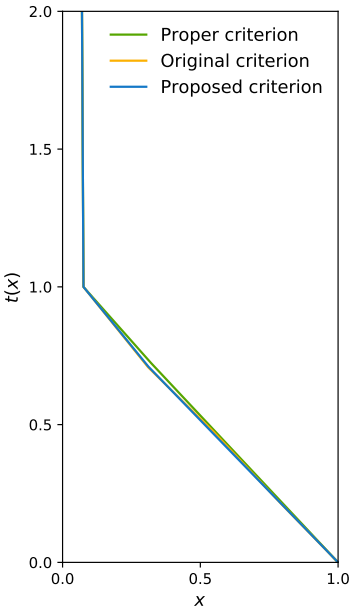

Figure B.6. $t(x)$ for $T^{\mathrm{F}}_{10^{5}}$, $|P| = 50$, $k = 5$

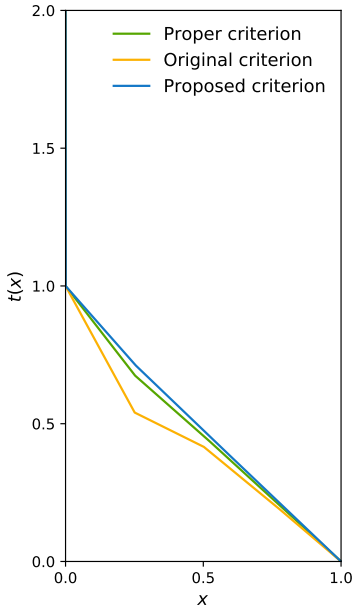

Figure B.7. $t(x)$ for T_2^{Y}, $|P| = 5$, $k = 5$

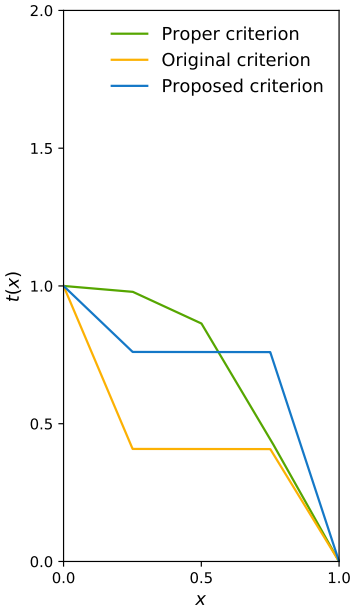

Figure B.8. $t(x)$ for T_2^{Y}, $|P| = 50$, $k = 5$

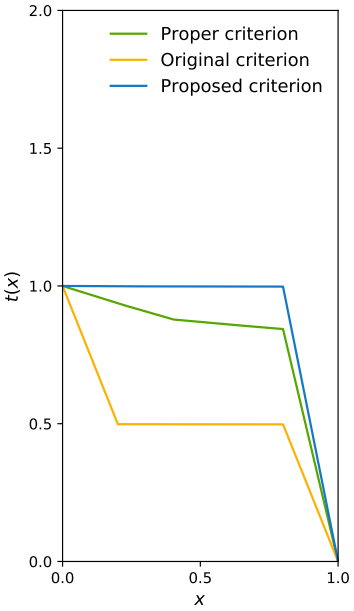

Figure B.9. $t(x)$ for $T_{0.3}^{\mathrm{Y}}$, $|P| = 50$, $k = 5$

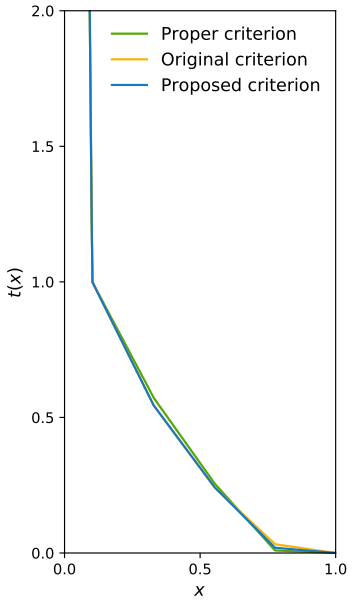

Figure B.10. $t(x)$ for T_2^{Y}, $|P| = 5$, $k = 5$

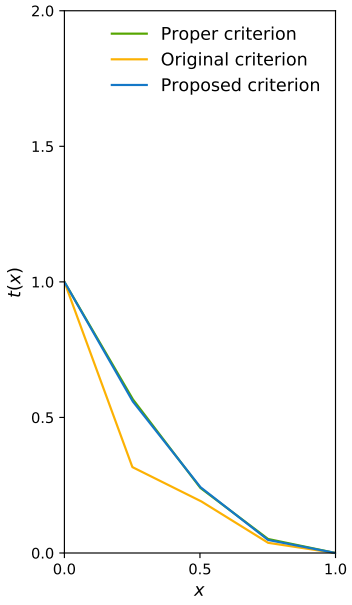

Figure B.11. $t(x)$ for T_2^{Y}, $|P| = 50$, $k = 5$

Appendix C
Selected predicted vs. observed plots

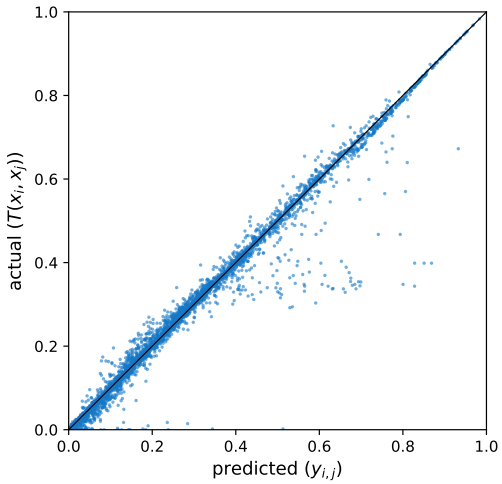

Figure C.12. Original criterion, T_{P}, $|P| = 5$, $k = 5$

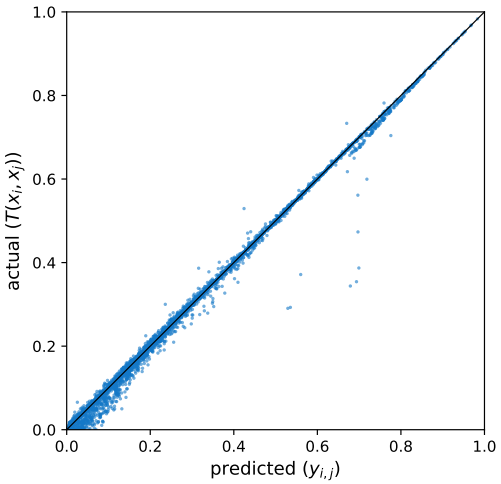

Figure C.13. Proposed criterion, T_{P}, $|P| = 5$, $k = 5$

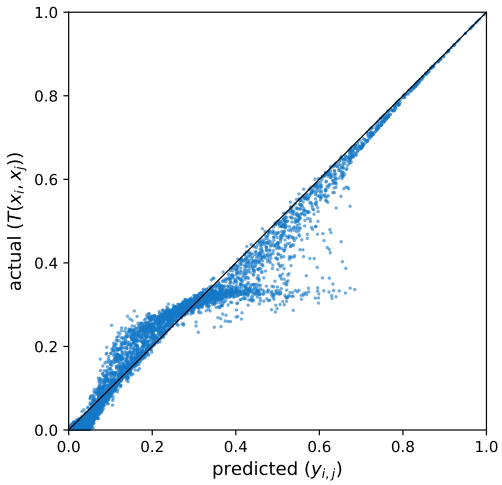

Figure C.14. Original criterion, T_{P}, $|P| = 50$, $k = 5$

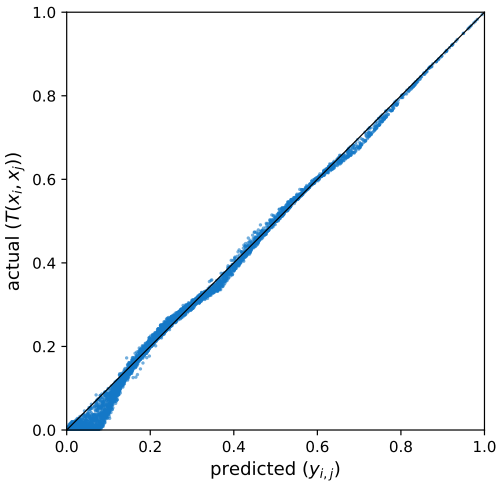

Figure C.15. Proposed criterion, T_{P}, $|P| = 50$, $k = 5$

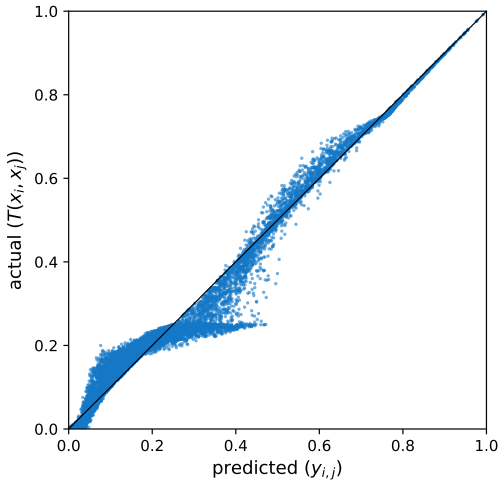

Figure C.16. Original criterion, T_{P}, $|P| = 100$, $k = 5$

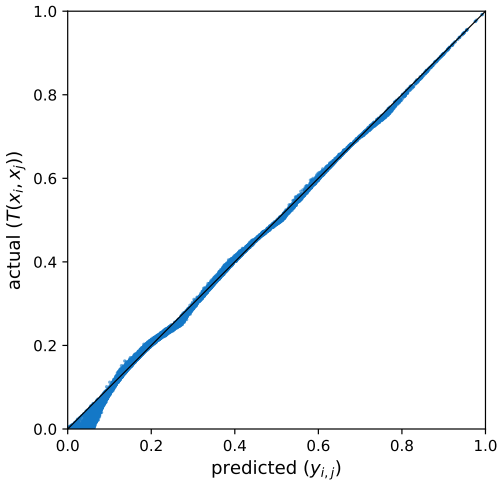

Figure C.17. Proposed criterion, T_{P}, $|P| = 100$, $k = 5$

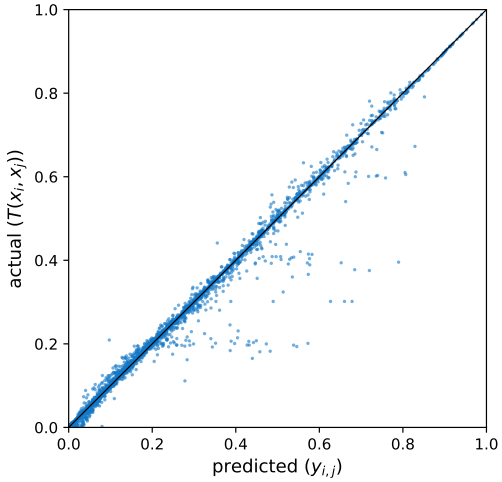

Figure C.18. Original criterion, T_{P}, $|P| = 8$, $k = 7$

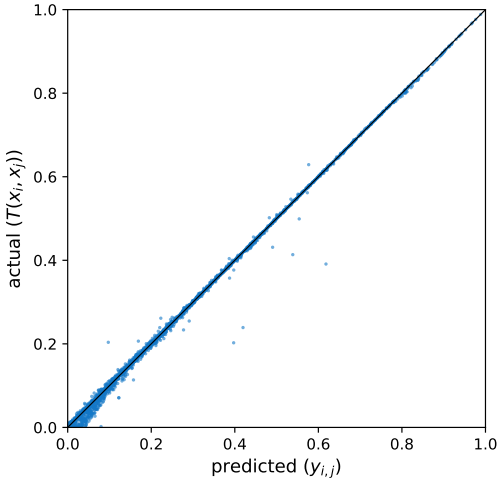

Figure C.19. Proposed criterion, T_{P}, $|P| = 8$, $k = 7$

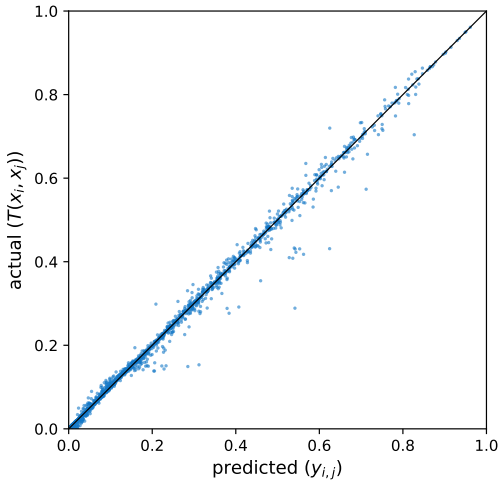

Figure C.20. Original criterion, T_{P}, $|P| = 15$, $k = 9$

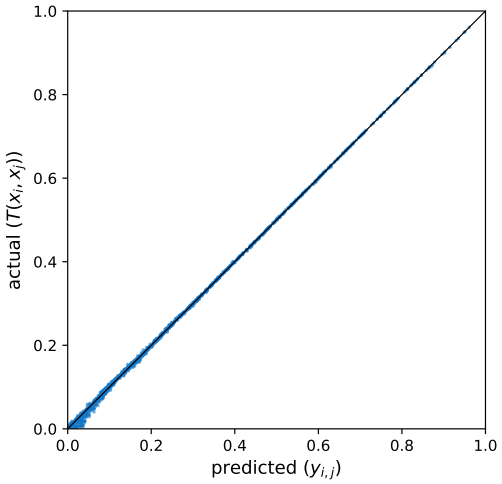

Figure C.21. Proposed criterion, T_{P}, $|P| = 15$, $k = 9$

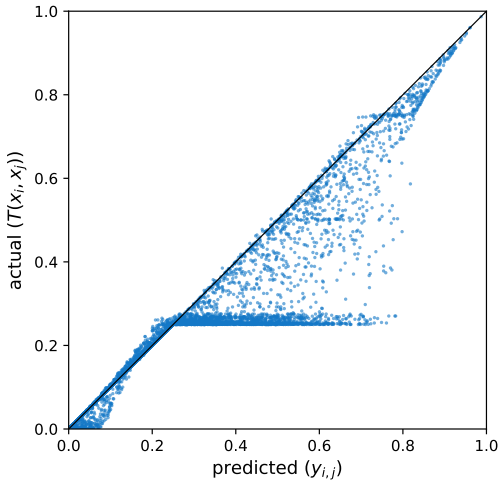

Figure C.22. Original criterion, $T^{\mathrm{F}}_{1\cdot 10^{-5}}$, $|P| = 50$, $k = 5$

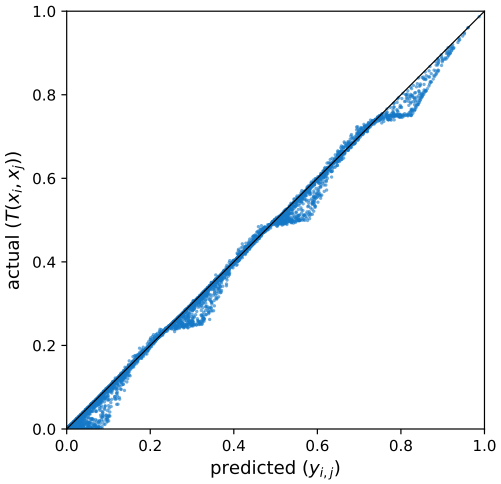

Figure C.23. Proposed criterion, $T^{\mathrm{F}}_{1\cdot 10^{-5}}$, $|P| = 50$, $k = 5$

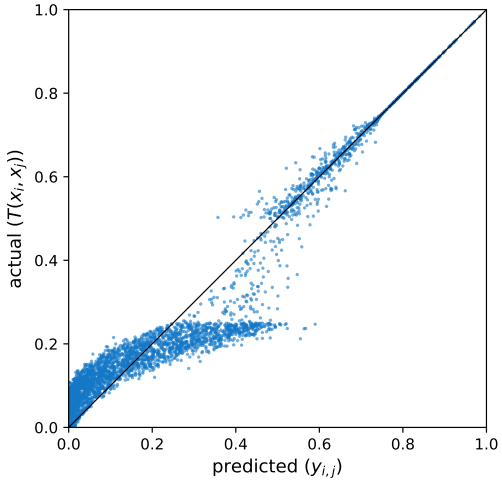

Figure C.24. Original criterion, $T^{\mathrm{F}}_{1\cdot 10^{5}}$, $|P| = 50$, $k = 5$

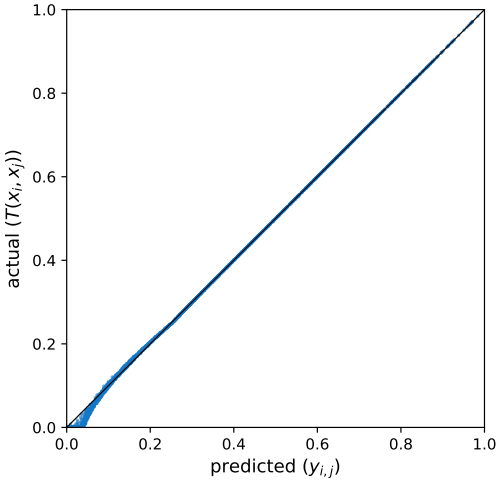

Figure C.25. Proposed criterion, $T^{\mathrm{F}}_{1\cdot 10^{5}}$, $|P| = 50$, $k = 5$

Appendix D
CD content

The content of the CD attached to this thesis is listed in Table D.1. Instructions on how to run the code are included in the `README.md` file inside the `source_code` directory.

Directory name	Content description
text	This thesis in the PDF format
source_code	Code used in the experiments

Table D.1. CD content