Whitepaper

PANDA | METRIC - an Open Source Framework for Industrial AI Applications

Michael Welsch

PANDA | METRIC is a framework for processing any data. It extends the capabilities of machine learning algorithms for variable structured data types and enables statements to be made about the relationships between the data in the context of artificial intelligence. For this purpose, the recorded data is first modeled using a metric in a metric space. These spaces can now be related to each other and simplified without loss of information. This allows essential information to be extracted and accessible to the user. In various modules, the framework offers a collection of algorithms that are optimized for metric spaces and accessible via a standardized API. PANDA | METRIC is programmed in modern and template-based C++, which allows a comfortable use with optimal performance at the same time. Compared to the approach of neural networks, the concept of metric spaces offers significant advantages for industrial applications.

I. The metric space

A metric space is a set to which a function applies that defines a concept of similarity between any two elements of that set.

For example, a set can consist of 1 million recorded images from a camera. The distance function or the metric now provides a numeric value for any two elements of this set. The more different the two selected images are, the larger the numerical value is, the more similar they are, the stronger it tends towards zero.

Formally, a metric space is already implicitly created by allocating a suitable metric to a set. Within the algorithms, the set is now usually explicitly 'multiplied out' using the metric. This can happen e.g. by a complete pairwise calculation and storage of the distance values in a distance matrix. However, due to its quadratic complexity, this approach quickly reaches the limits of computer technology.

Alternatively, the metric space can be represented in a sparse graph or in a search tree, see Figure 1. Both tree and matrix can be interpreted as special cases of a general graph concept for the explicit representation of a metric space.

The edges in this graph represent the distances between the individual elements. Each of the three principles for the representation of a metric space has certain advantages and disadvantages. Accordingly, these principles are used differently in the algorithms. Some of the distances are also calculated dynamically.

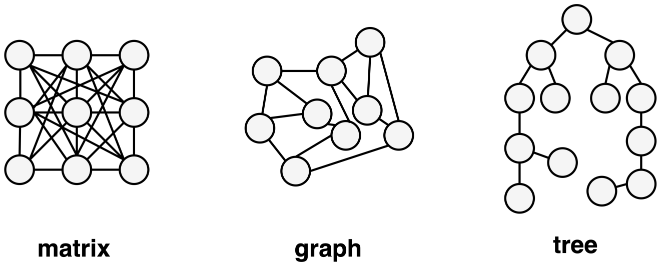

Figure 1. Representation of the Metric Space

A. The Metrics

The technical literature contains a large number of different metrics. The Euclidean metric is best known as the distance between two points of a geometric space along a connecting straight line. In addition to the Euclidean metric, there are a number of related metrics that are also well known in classical machine learning, such as the City-Block metric. However, there are many more metrics, such as for x-y curves, images, texts, or stochastic sampling. There are also metrics to calculate the distance between two metric spaces.

The execution of the metric concept in Wikipedia indicates that the metric concept is basically (only) a generalization of the Euclidean metric. However, this version is insufficient. Generalized, a metric corresponds to the minimum cost of an information-theoretical recoding of one element of the set into another element of the same set.

The conversion process can and usually is hindered by constraints, which is why every metric can be generalized with the mathematical concept of linear programming.

While the Euclidean and related distance functions can be calculated explicitly between two elements, for more complex data types there is generally an optimization problem for the calculation of the distance. This is due to the fact that there are many possible ways of recoding, but only the way with the lowest cost corresponds to the distance searched for.

In the case of an image comparison, for example, a metric can be selected which, analogous to the Euclidean concept, simply compares pixels for pixels or implements a metric with object recognition. The optimization algorithm can, for example, transform the image of a dog into the image of a cat and determine the minimum costs for the pixel conversion. These can then be larger than the costs for the conversion of a cat into another cat. The metric in the framework used for this is based on graph modeling, while the optimization problem of simpler data types such as x-y curves is solved by dynamic programming.

In the METRIC framework, the different metrics are divided into *k-random*, *r-related* and *k-structured*.

Practically every element is defined by a vector from individual entries or provided in the framework by an accessor function interface in order to directly access arbitrarily structured data in databases. In the case of images or higher dimensional constructs, the entries are serialized and, for example, the constraints of height and width of an image are only supplemented via the metric or the *Metric Factory Function*.

With a k-related metric, the order of the k entries in the array can be principally swapped without affecting the result of the calculation, whereby the swapping must occur equally in all arrays. With a k-random metric, the order can also be exchanged arbitrarily within each array.

With k-structured metrics, however, the order has a structural meaning (as in images, curves and sentences), which is why the position of the entries in the metric is explicitly taken into account.

The user thus brings in a certain a priori knowledge about the data by choosing the metric. If this is not possible, the user can have the constraints of the metric calculated automatically or test whether there is a k-random, k-related or k-structured metric in general, in order to then select a corresponding metric.

II. Metric algorithms

The algorithms in the METRIC framework are organized into several submodules. **SPACE** provides classes for the representation of metric spaces including basic operations such as the search for neighboring elements. **DISTANCE** provides an extensive collection of metrics, including factory functions for configuring complex metrics. The module **CORRELATION** offers functions to calculate a correlation coefficient of two metric spaces and thus to determine the dependency of two arbitrary data sets. In addition to the arbitrariness of the elements, another advantage over classical methods is that the metric correlation is nonlinear. Thus it does not show the typical misinterpretation of a Pearson correlation. The correlation algorithm works with a subspace sampling approximation of the metric space for data sets of any size. The **TRANSFORMATION** module contains deterministic algorithms that do not map a new metric space out of the structure of a metric space, but transfer element by element into another metric space, e.g. from the time to the frequency domain. A distinction can be made between lossy compression and completely reversible methods. In general, however, these are proven, deterministic methods such as wavelets.

The **UTILS** module contains algorithms which are not metric either, but which can be easily combined. On the one hand, there is a high-performance in-memory crossfilter. This allows a piecewise, UI supported and interactive filtering of the patterns from the results of the operations in real time. On the other hand, the UTILS module also contains a nonlinear and *nonparametric signifcance test* for independent features (PMQ) of a metric space that were obtained by mapping.

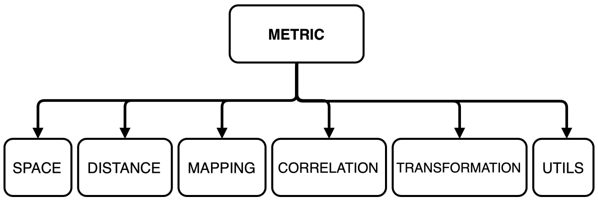

Figure 2. Structure of the METRIC framework

MAPPING contains various algorithms that can 'calculate' metric spaces or map them into equivalent metric spaces. If, for example, a camera takes pictures of a scene in which one door opens and closes. If this additionaly always happens under different lighting conditions, the set of elements initially receives millions of extensions analogous to the number of pixels. All pixels differ from image to image. In reality, however, there are only two degress of freedom: the opening angle of the door and the brightness of the light source. A mapping algorithm can for instance now calculate these features and transfer the millions of dimensions of the original metric space to a metric space with only two dimensions per element.

Images can also be bidirectionally reconstructed from these two features using the reverse decoder. Since these two calculated features are now relatively independent, whereby this can also be checked with a metric algorithm, complex data can be interpolated by means of these images and back-images under consideration of the constraints. In addition, unwanted features, such as the influence of brightness, can be removed from the images for further evaluation. In this way, the user brings in his a priori knowledge and understands on the other hand what his a priori influence causes instead of causing a loss of information by an autonomous programming of this knowledge.

For example, the user knows a priori in an FFT analysis that oscillations are relevant, but separates the overtones in the transformation, although this does not correspond to the human transient perception. A metrically based mapping in a feature space, on the other hand, leads to the oscillation patterns in the time domain including overtones being isolated and allowing a substantially more intuitive interpretation.

III. ARTIFICIAL INTELLIGENCE SKILLS

The algorithms in the MAPPING module aim to reduce complexity with clustering or classification and to fill missing information in the data. This is a prediction that is an essential part of machine learning for numerical data in tabular form and is now also available for any data type and data combination. On the other hand, the algorithms in the MAPPING module can also be used to restructure data or to merge different metric spaces and thus different data sources.

This can be used, for example, for synthesizing abstract images from any data source automatically. All these methods are not based on the labeling of data by the user and cannot be interpreted as learning. METRIC is therefore not a machine learning tool in its core, although it is much more familiar with this sub-discipline of artificial intelligence (AI) than other frameworks.

In addition to the ability to link complex information and recognize patterns without extensive and redundant pre-training, most people understand AI as the ability to make decisions independently and act automatically. This is generally feasible with the metric concept by interpreting an action strategy as a metric and defining a metric action space. As with any other metric, there is an optimization problem under many constraints. Features in this case would be e.g. meta-production strategies like quality, cost and time. An optimal state can be achieved automatically by mapping back individual meta-production parameters and by navigating within the metric space by generating new data points. In the first version of METRIC, however, this goal is not (yet) pursued, since autonomous action would meet with rejection, unresolved liability issues and long certification. Even if the algorithmic or mathematical step is very small, more than 99% of the necessary data of smart components as well as control variables are missing in every current production plant for such an AI functionality.

Generally, by mapping the data in metric spaces, software is able to capture the entire knowledge in the data completely and simultaneously, which is something humans are unable to do with their capabilities. A camera at the last station of a production line can be listed here as an example. Its pictures are represented completely and with all its details in the metric space, but the user is only interested in a certain influence on the pictures and not in all of them, because the abundance of unfiltered information would be overwhelming for him.

The first version of METRIC in combination with the **Drift UI** is about reducing this complete knowledge in the data to a level that helps people to understand an industrial process and to optimize it themselves. The millions of dimensions are simplified to a 2D representation and thus reduced to the human measure of processability in order to achieve a focused and selective view of the overall context.

This is done via a UI concept within the drift frontend and an interactive workflow, so that the user consciously links his a priori knowledge about the data bit by bit.

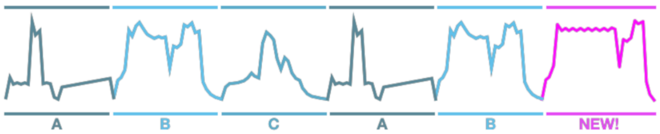

Figure 3.: Intelligent Pattern Recognition

The user is always aware of the consequences of his or her influence until he or she receives a message that is understandable to him or her. Eventually, the aim of this assistance system is to obtain answers to questions regarding certain data, for which additional a priori information from the user is usually required. With increasing sensor information density and actuator options, the next stage of expansion will see a real autonomy of the system beyond a mere assistance function.

Industrial AI is not an advertising slogan for PANDA, but a serious technological promise. The basis for this lies in the metric mathematics being used. By means of the first version of METRIC and the UI of Drift, two simple questions are to be answered by the AI: What goes wrong and why?

The AI will always ask the user to show it what is normal for them by example. For instance, the user defines an early shift as the reference time value for good data and filters out 5% of the most anomalous values.

In order to provide the user with this first level of the AI, the practical challenge lies in the combination of the algorithms in the METRIC framework accordingly. For example, the features obtained through mapping can be used as a label for a classification to check which features cause anomalies in the overall construct or correlate with other data sources. Subsequently, the most significant feature dependencies are modeled and visually processed, see Figure 3.

For a specific process curve, for example, the start-up overshoot when the tool is positioned could be something that correlates with the subsequent plateau and only a change in ratio could be an actual feature. The question arises whether this phenomenon exists as a drift or anomaly and is processable. In this way, the user is playfully given a new understanding of his process. Through an interactive retransformation of the data, the user can gain exactly this knowledge, understand his process better and thus trust the technology.

IV. Data compression

As a side effect, the metric concept offers data compression that goes well beyond state-of-the-art compressors. The complex data is first decomposed by a wavelet transformation. By encoding the 'oscillating' redundancies, the arrays remained after the noise removal are sparsely filled, which greatly accelerates the metric calculation. Afterwards, only the patterns are stored in metric spaces and mapped by features. The sequences of the patterns are also represented by a metric in metric spaces. The AI tries to estimate which pattern comes next. If the prediction is correct, no more storage space is needed to obtain the information. The data is then completely reconstructed from the pattern and the sequences, which in turn are patterns. Since the processing inevitably takes place via metric spaces, the original raw data is not stored, since it can simply be synthesized if necessary.

The METRIC framework incorporates many individual scientific findings that make common sense through the framework. A list of selected publications can be found in the appendix of the paper available for download [1-10].

There are other mathematical principles that are compatible with the metric concept, such as *kernel* and *graph-laplacian* based methods. Selected graph-based analysis methods also find their way into the framework due to the mapping of the metric space as a graph. This can be used, for example, to de-noise metric spaces and thus arbitrarily complex data. Furthermore, redundant information in a metric space can be eliminated, so that an exemplary number of 10 images from 1 million images can be provided as representations.

Furthermore, an assistant function can be offered to the user by enabling a completely self-learning clustering by some important labels when completely marking a data set, if the user needs this e.g. for a downstream analysis. The metric concept allows the user to get a complete overview of any number of data without having to explicitly view and visualize each individual data set.

V. DIFFERENCE FROM METRIC SPACES TO DEEP LEARNING APPROACHES

Neural networks usually involve the training of f() in y=f(x) with the help of x and y, in order to calculate a predicted y from unknown x. These are high-dimensional and non-linear fitting functions, which should fill missing data. However, it is not answered what the 1% most strange elements look like and whether there are structurally not explainable anomalies or simply extremes of the 99% remaining elements or whether there is a connection with other data. In addition, such a network reacts to unknown input x with partly unsharp y and thus makes the procedure appear in many eyes to be inaccurate or unusable, especially when it comes to dealing with unknown and new data. In manufacturing, however, these unexpected phenomena are exactly what it is all about. Furthermore, the training of neural networks, especially with many layers including convolution layers, is very computation-intensive. It is also susceptible to numerical phenomena that can be traced back to the concept of backpropagation and can lead to useless solutions. To have these effects under control requires a very large engineering know-how and cannot be improved at the moment reasonably or only with exorbitant computational use by a metaparameter optimization. As a result, only pre-trained models are increasingly used, although this does not satisfy the specific needs of every data source in the industry. In general, the metric concept is based on *global optima*, while neural networks are always implemented with the best possible *local optima*, which can only be achieved with a disproportionate amount of computing effort. In contrast to the metric concept, the application possibilities of neural networks are limited.

VI. Open Source Software

METRIC is programmed in C++ 17 as header only and published under MPL 2.0 open source[1].

METRIC runs under Linux, macOS as well as Windows. All functions and classes in METRIC have a uniform API. The template parameters provided are always the record type and the metric type, which ensure that the compiler always produces the optimal machine code. Parameters are integrated via factory functions and an accessor interface bridges the gap between database modeling and the array format for the algorithms.

The algorithms are connected to the middleware and the frontend via an intermediate language in the form of Json-based communication. Since the application of a JIT compiler led to difficulties in debugging and to a virtualization, which goes strongly at the expense of the performance, the Template-Resolving system proved itself. Although this has disadvantages in compilation time and the resulting program size, it does not cause any problems in debugging and performance.

Literature

[1] E. Bridgeford, Censheng Shen, Shangsi Wang, und J. T. Vogelstein, „Multiscale Graph Correlation," 2018. [Online]. Available: https://zenodo.org/record/1246967

[2] A. Lulli, E. Carlini, P. Dazzi, C. Lucchese, und L. Ricci, „Fast Connected Components Computation in Large Graphs by Vertex Pruning," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, Nr. 3, S. 760–773, Mar. 2017. [Online].

[1] https://github.com/panda-official/metric

[3] S. Arora, S. Rao, und U. Vazirani, „Expander flows, geometric embeddings and graph partitioning," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing - STOC 04*. ACM Press, 2004. [Online]. Available: https://doi.org/10.1145/1007352.1007355

[4] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, und H. Bunke, „Approximation of graph edit distance based on Hausdorff matching," *Pattern Recognition*, Vol. 48, Nr. 2, S. 331–343, Feb. 2015. [Online]. Available: https://doi.org/10.1016/j.patcog.2014.07.015

[5] O. Pele und M. Werman, „Fast and robust Earth Movers Distances," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, Sep. 2009. [Online]. Available: https://doi.org/10.1109/iccv.2009.5459199

[6] S. T. Roweis, „Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, Vol. 290, Nr. 5500, S. 2323–2326, Dez. 2000. [Online]. Available: https://doi.org/10.1126/science.290.5500.2323

[7] D. A. Spielman und N. Srivastava, „Graph sparsification by effective resistances," *Proceedings of the fourtieth annual ACM symposium on Theory of computing - STOC 08*, 2008. [Online]. Available: http://dx.doi.org/10.1145/1374376.1374456

[8] C. Jiang, H. Xie, und Z. Bai, „Robust and Efficient Computation of Eigenvectors in a Generalized Spectral Method for Constrained Clustering," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Serie Proceedings of Machine Learning Research, A. Singh und J. Zhu, Hrsgg., Vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, S. 757–766. [Online]. Available: http://proceedings.mlr.press/v54/jiang17b.html

[9] M. Hein und M. Maier, „Manifold Denoising," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, und T. Hoffman, Hrsgg. MIT Press, 2007, S. 561–568. [Online]. Available: http://papers.nips.cc/paper/2997-manifold-denoising.pdf

[10] M. Izbicki und C. Shelton, „Faster cover trees," in *Proceedings of the 32nd International Conference on Machine Learning*, Serie Proceedings of Machine Learning Research, F. Bach und D. Blei, Hrsgg., Vol. 37. Lille, France: PMLR, 07–09 Jul 2015, S. 1162–1170. [Online]. Available: http://proceedings.mlr.press/v37/izbicki15.html

[11] „Wavelet Data Compression." [Online]. Available: https://www.mathworks.com/help/wavelet/ug/wavelet-data-compression.html