András Benczúr

Data-driven Methodologies and Big Data

András Benczúr PhD, Head of the Informatics Laboratory of the Institute for Computer Science and Control of the Hungarian Academy of Sciences; Leader of the "Big Data – Momentum" research team

Abstract

Traces of human activities, communication, travel, work can all be found in information systems. Data from these systems can be analyzed by machine learning tools that identify the patterns of normal behaviour and identify unusual or risky events from continuously produced streams of data.

In our study, we present the possibilities of data-driven Big Data analysis to detect security risks. Machine learning methods work proactively: instead of triggering the investigation by a certain event or query, all data are permanently monitored, with patterns and models continuously being applied to identify potential signs of risk.

During the over 20 years of their use, data mining methods have changed trends partly because new methods have appeared, and partly because knowledge from actual results have explosively increased. Early textbooks focused on clustering and association analysis, while currently new methods of classification such as boosting and deep learning are gaining ground.

Finally, we present the main quality of Big Data, which lies in the fact that traditional methods cannot cope with the computational requirements for data analysis. We introduce the Big Data phenomenon, the challenges and emerging answers. Recent, mostly open-source distributed software systems are capable of utilizing a large number of cheap commodity servers to implement complex data preparation and analysis tasks, which makes it possible, for example, to detect risk patterns in Web-scale social media streams.

Keywords: anomaly detection, Big Data, data mining, data-driven methodology, classification, data science, machine learning, models of security

Introduction

Most aspects of our daily life are already linked to information systems: almost all of the events of telecommunication, web, social media, transport, travel activities, and monetary transactions are recorded in the log files of servers, and in certain cases – for example, the

content of web and social media – they can be accessed by anyone. When we are looking for risks, we can obviously leverage all the data available to us. For example, we can link names in criminal records to those retrieved from social media, and we can map in this way the circumstances of known abuses, and we can extract the characteristics of the persons involved. In a similar way, we can launch an investigation based on public content discovered on the web when we need to find the actual persons behind some content that poses a risk. However, we can also look at the data as a whole in which known persons, suspicious content and relationships can be transformed into attributes and huge tables that contain quantitative information and can be used for model building, allowing us to look for known or yet to be discovered risk patterns.

The key characteristic of data-driven surveillance is that it is proactive instead of being triggered by an event. A machine learning process will assign a risk classification to all events arising in the system, contrary to search-based surveillance that operates by retrieving the environment and the characteristics of specific events and persons, looking for similar objects. When searching, we will probably identify the patterns only in a very small portion of the data. Currently, the most widely accepted name for the discipline that develops predictive models based on available data is data science.

Certain predictive modeling methods, such as the decision tree (Safavian et al., 1998) or logistic regression (Cox, 1958) have been used in statistical analysis for more than fifty years. The phrase *"data mining"* emerged in the middle of the 1990s to identify the discipline that puts emphasis on the issues of size and efficiency in data analysis (Han et al., 2001). One of the first innovations of data mining was associative analysis, which, however, along with a few other methods, proved to be of limited use in practice.

Today data science primarily focuses on the processing and cleansing of huge data volumes, the generation of attributes, followed by the learning of the attributes of individual data points. It employs new methods, such as deep learning which has recently become computationally feasible with the improvement of the algorithms for training large neural networks, or the so-called boosting procedures that have extremely beneficial properties for practical applications. About ten years ago it was mostly the explosive growth of web-based data volumes that drove the emergence of *big data* as a distinct discipline, which has become more of an interdisciplinary field supporting data science and offering recommendations for solutions to special data volume related issues. The most important tools of big data are open source software applications like Hadoop (White, 2012), Spark (Zaharia, 2010) or Flink (Carbone at al., 2015), all engineered by Apache Software Foundation.

This study is intended to describe how data science methodologies can be used to discover security risks. We will also give an overview of the methods that are an integral part of data science and incorporate practical experience and new scientific advances starting with the first procedures that have been in use for more than fifty years. First we will introduce the two major types of security systems – the positive and negative model – that are capable of recognizing risks and we will examine their advantages and shortcomings. In the next subsection, the principles of data-driven methodology and the methodology of providing training and evaluation test data will be described. It will be followed by a discussion of the new trends in machine learning, the factors behind the changes, and the features of the new methods that are most important in applications. Finally, we will examine the phenomenon of big data and touch upon the methods of analyzing, for example, the data deluge pouring from the social media to recognize risk factors.

Security models

The significance of risk models lies in the fact that they describe potential behaviour patterns that we wish to identify and select from a set of possible events to conduct an investigation or to take preventive action. The concept of risk models originated from the network security field of information technology (Bertino et al., 1997), but obviously it is also present in any other types of security systems. The first risk models were based on positive authorization patterns that were later supplemented with negative and mixed rules (Bertino et al., 1997) (See Figure 1). Next follows the explanation of the essential differences between the two types of models in terms of their potential applications as well as their relationship to machine learning methodologies, and we will also address the possible ways to move forward and combine the best of both worlds.

This paper will focus on machine learning mechanisms implementing the negative security model. We find it very important to explore the extra capabilities of the positive model and the alternative methods that can be used to extend typical negative models.

Negative security model

A negative security model built of past security events that represented threats can only detect security events that have a precedent. This solution is quite efficient if the number of detected security events is significant and their characteristics do not change over time, which makes these systems better suited for catching credit card fraud than detecting acts of terrorism.

Positive security model

We seldom encounter positive security model based solutions, especially one that can handle data from the transactions, the computer network and the physical sensor network in a standardized way. A key part of the possible solutions is to analyze hidden relations in the data, for example by finding identical attributes between different entities (e.g. the same home address shared by "different" clients).

Further directions

In the past ten years, machine learning applications have mainly focused on supervised models. This means that current systems are mostly based on the negative security model where some anomalous events had to be detected and logged previously in order to train the system. Use of the positive security model may be enabled, for example, by the so-called active learning methodology (Tong et al., 2001), where the system constantly requests expert feedback about events it is uncertain how to classify. In this way human collaboration makes it possible to investigate cases suggested by the machine, leading to the formulation of a model tweaked for more accurate automated recognition.

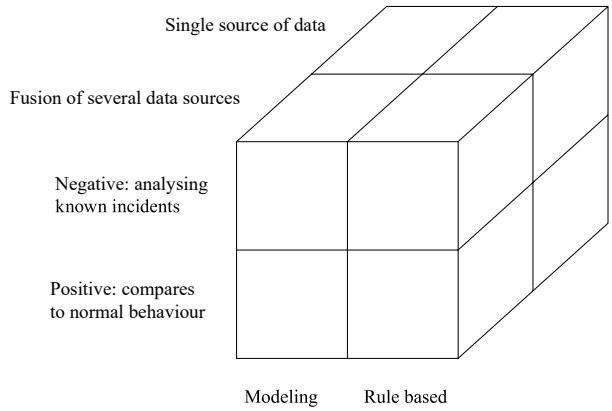

Figure 1

Security models

Source: The author's own contribution

Figure 1 shows the central point of the data-driven methodology in a three-dimensional spatial diagram of the security models. Contrary to rule-based systems that require tremendous manual work, data driven methods learn patterns from the data. While most of the conventional systems check data sources one by one, data driven methods can model the joint behaviour of several data sources.

Data science and data-driven methodology

The data-driven methodology draws conclusions by processing and analysing all available data, and by using qualitative methods. The possible methods can parse huge amounts of various types of data (text, physical measurements, data traffic, date, period, monetary amount or category). Operation of data-driven systems is shown in Figure 2. After cleansing the data and converting them to a uniform format, classical or more recent data mining and machine learning algorithms are used to create patterns, signatures, extracts, profiles and aggregates for further analysis and modelling.

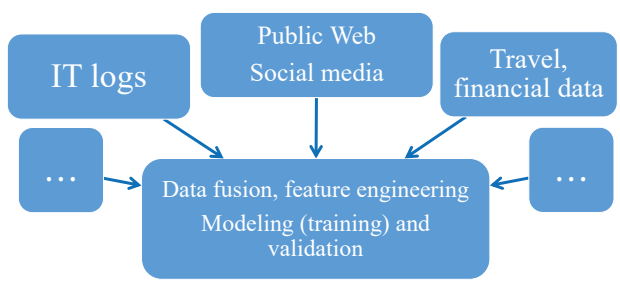

Figure 2

Central data fusion phase of the data-drive platform

Source: The author's own contribution

In the phase of generating the attributes, we need to select the base object of prediction, which can be a person, an IT system event, or a financial transaction. Available data about the base objects should be transformed to produce an array of attributes associated with the objects as rows. Figure 3 shows numeric and categorical values. Figure 4 shows a table consisting of text data.

ID	Activity Month 1	Activity Month 2	Activity Month 3	Travel to monitored target location	Risk at place of stay	Distance in network from suspects	Physical distance from suspects	…	Risk (label)	M1 (logistic regression)	M2 (depth 2 tree)	M3 (boosted tree)
1	1000	1100	1200	0	0.2	…	…	…	no	0.17	no	3
2	1000	1200	1100	1	0.5	…	…	…	yes	0.49	no	-1
3	2000	3100	3200	1	0.1	…	…	…	no	0	no	1
4	2000	2200	2100	0	0.3	…	…	…	no	0.39	no	3
5	3000	3100	3200	2	0.6	…	…	…	yes	0.84	yes	-3
6	3000	3200	3100	0	0.5	…	…	…	no	0.54	no	1
7	3000	4100	4200	1	0.2	…	…	…	no	0	no	1
8	3000	3200	3100	3	0.3	…	…	…	yes	0.96	yes	-1
9	4000	5200	5100	0	0.3	…	…	…	no	0	no	3
10	4000	4200	4100	0	0.7	…	…	…	yes	0.69	yes	-1

Figure 3

Sample data of ten suspected persons, with numerical attributes

Source: The author's own contribution

(1) John likes to watch movies. Mary likes movies too.
(2) John also likes to watch football games.

List of words: "John," "likes," "to," "watch," "movies," "also," "football," "games," "Mary," "too".

Word bag representation: (1) [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
(2) [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

Figure 4

A set of text data examples and their bag of words representation

Source: https://en.wikipedia.org/wiki/Bag-of-words_model

The output of data-drive methodologies is the prediction associated with the individual objects that can be binary (yes/no), a risk probability, or an estimated risk value. We can measure the performance of the system either by new events entering the system or by using the test data set compiled during model building. We will explain the training and validation options and certain evaluation metrics below.

When training the model, we need to compile information from previous events and use them as *training data* for building our model. The completed model can be applied to

making predictions based on the constant influx of new so-called *test data* or the *validation data* that was set aside at the time of building the model. Test data can be selected from what is the future relative to the moment of building the models, or by sampling the available data. Cross validation is a frequently employed method where objects are randomly divided into several groups, and then one of these groups is selected in every possible way as a validation set to calculate the average performance of the produced models. Measuring model performance on training data is not recommended because we will overestimate the quality. We may also overfit the model, which means that there is an alternative model that underperforms on the training data, but outperforms the original model when it is applied to the test or validation data set. Overfitting restricts our ability to generalize the model to unknown cases. The possible training and validation methods are summarized for example by Tan (2013, Section 4.5).

Finally, there are two options to quantify the quality of predictions. If the method in question produces a binary (yes/no) risk forecast, the quality can be derived by reviewing the so-called class confusion matrix (Power, 2011). The four elements of the matrix (as shown in Figure 55): the numbers of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) events.

	Actual risk: YES (positive)	**Actual risk: NO (negative)**
Classified as risk by M1: YES (positive)	True Positive (TP) 3 (ID 5, 8, 10)	False Positive (FP) 1 (ID 6)
Classified as risk by M1: NO (negative)	False negative (FN): 1 (ID 2)	True Negative (TN): 5 (ID 1, 3, 4, 7, 9)

Figure 5

Confusion matrix for risk prediction for threshold value M1 > 0,5 of model M1 in Figure 3 Accuracy in the example: (3 + 5)/10 = 80%; precision: 3/(3+1) = 0.75; recall: 3/(3+1) = 0.75; false positive rate: 1/(1 + 5) = 0.16.

Source: The author's own contribution

From the confusion matrix, the following values can be derived:

- accuracy: (TP + TN)/total number of cases,
- precision: TP/(TP + FP),
- recall or true positive rate: TP/(TP + FN),
- false positive rate: FP/(FP + TN).

If we examine a model that assigns risk probabilities to the individual incidents, then additional quality measurement options will be available. We can obviously create a confusion matrix for any risk thresholds: entries below the threshold will have a negative, those above the threshold will have a positive classification. We will get a more stable measure, however, if we manage to evaluate the prediction independent of the threshold value.

The so-called *ROC (Receiver Operating Characteristic)* curve (Swets, 1996) plots the true positive rate for the possible threshold values in related to the false negative rate (Figure 6). The area under the curve is called *AUC (Area Under the ROC Curve),* which is a value between 0 and 1. The AUC value determines the probability that a random positive

event is ranked higher than a random negative event by the model. AUC = 1 indicates perfect prediction, while AUC = 0.5 means that the model is guessing randomly. If the AUC values are below 0.5, we will get better results by inverting the output of the model. As a result, we can work with the Gini = 2 × AUC – 1 value.

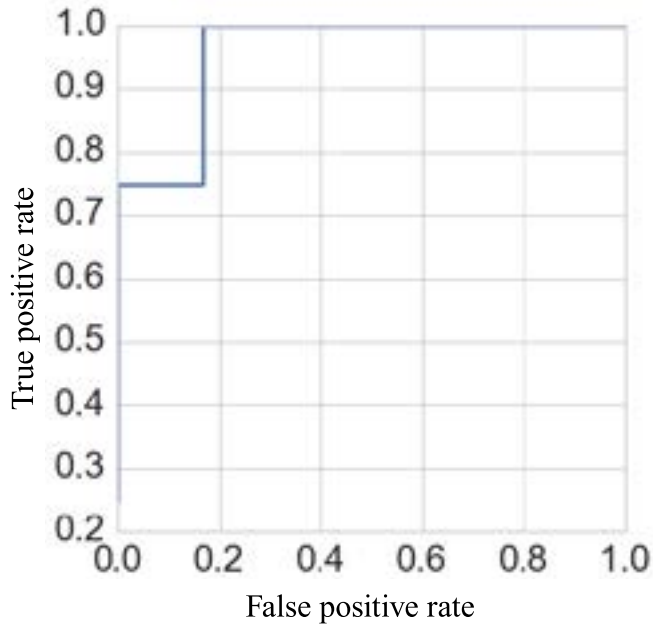

Figure 6

AUC curve for prediction M1 in the example of Figure 3

Source: The author's own contribution

Supervised and unsupervised learning

Supervised learning relies on training data labelled by an expert or as a result of some process. Supervised machine learning is best suited for implementing negative security models as it relies on known risk events. The unsupervised methods do not require the manual labeling of data, and as such they are perfectly suited for positive security models in theory. However, the practice of machine learning shows that it is very hard to use supervised learning in real-world environments as it is difficult to evaluate the quality of the methods and interpret the results.

In this subsection, we will introduce the most important supervised classification, regression and anomaly detection methods, and we will also describe the limitations of the popular (unsupervised) clustering approach. We will not discuss frequent pattern and association rule mining, the earliest methods of data mining, as they did not prove to be useful in practice.

Supervised learning progressed considerably in the past twenty years. The earliest methods, such as decision tree (Safavian et al., 1998) or logistic regression (Cox, 1958), was followed first by *SVM (Support Vector Machines)*, which, for example, is excellent for classifying text entries (Joachims, 1998). Boosting-based methods (Freund–Schapire, 1997) are usually included in the solutions at most of the data analysis contests (Chen et al., 2016). The scope of application for deep learning (Lechun et al., 2015) has been expanding continuously beyond its original use, classification of images and videos.

In order to look for anomalies, most of the supervised classifiers can be transformed into a single-class version that receives only the elements of one of the classes (that represents normal behaviour) during the training. Anomaly detection, however, is characterized by the general issues of unsupervised methods: evaluation and interpretation are typically difficult. Very often, for example, we may find vast amounts of anomalous events that are not relevant in terms of security.

Classification

A prerequisite of classification is that events should be labelled automatically or through human intervention into classes. Classification is based on the negative security model in the first place, but it can also learn the patterns of normal operation. In the history of classification, after the initial methods (decision trees, logistic regression) there were quite a few paradigm changes which we will describe briefly in the following section along with an overview of the key methods.

Conventional methods

Next, we will explain the key principles of "conventional" classification methods already used before the millennium.

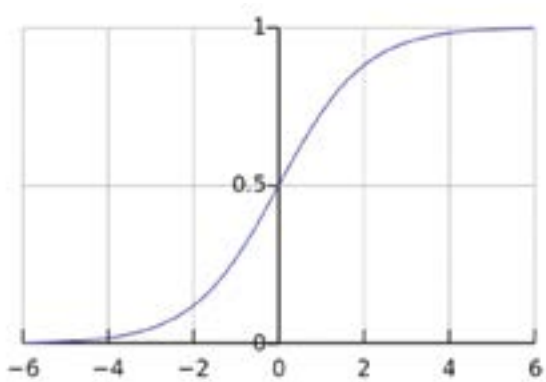

Figure 7
The form of the expit or logistic function

Source: https://en.wikipedia.org/wiki/Logistic_function

The input of *logistic regression* (Cox, 1958) consists of numerical variables. We learn the weight of each variable to calculate the weighted sum of each event. Above a threshold value a positive classification (risky) and below the threshold a negative classification is assigned to the given incident. When applied to the data set in Figure 3, the following formula will produce a 100% accurate model

score = expit (– 0.2 × Travel – 0.8 × Location + 0.5)

where the plot of expit or logistic function $\frac{1}{1+e^{-x}}$ is shown in Figure 7.

Logistic regression tends to overfit to strongly correlated variables. For example, the following formula obtained through actual training

score = expit (–0.0136 × Activity1 + 0.004 × Activity2 + 0.009 × Activity3
–1.0 × Travel –0.15 × Location)

reaches an accuracy of only 60%.

The name of the method is related to the use of the logistic function. The logistic function transforms the weighted sum of variables into a value between +1 and -1, and so it can assign a risk probability to every incident. The prediction that belongs to the first model in the example that uses only the Travel and Location variables is shown on the left side of Figure 8. The distribution of risk probabilities is shown on the right side. It is important to know that the threshold value applied to the weighted sum of the variables will always produce a high-dimensional linear separation of the data points similar to that on the left side of Figure 8, and thus it is not capable of recognising classes delimited by curves. This limitation will be overcome by the SVM method that we will describe in the following subsection.

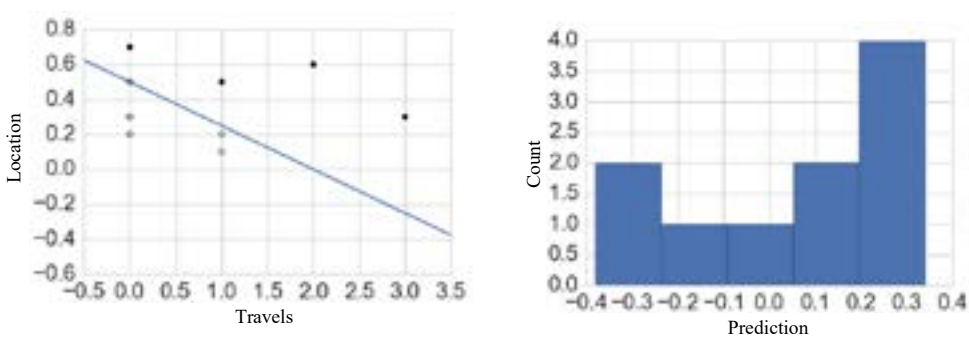

Figure 8

Logistic regression of data in Figure 3

On the left: separation of cases according to the model containing the Travel and Location variables On the right: distribution of risk probabilities produced by the model

Source: The author's own contribution

Logistic regression is very popular today. For example, in the financial sector it is a competitive modelling procedure typically used for making crediting-related predictions.

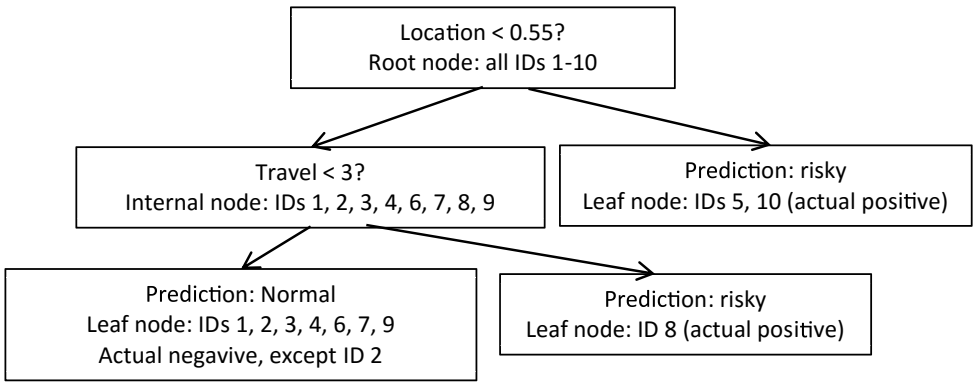

Figure 9

Depth two decision tree for the data in Figure 3

Source: The author's own contribution

As shown in Figure 9, *decision trees* (Safavian et al., 1998) split data points progressing from the root node (which is positioned at their upper end) by selecting variables with threshold values. As we proceed from top to bottom, we keep splitting the data set. While training the model, we need to find the decision that separates the two classes as much as possible in order to split the data points. If the quality of split is not satisfactory or we think the tree is too deep, we stop and construct a leaf node where the data points assigned to the leaf will determine the output decision as their majority class. Decision trees are popular due to the fact that the model is easy to interpret. Their quality, however, is lower than that of other methods. The renaissance of decision trees was brought about by the boosting methodology we will describe later, which can create high quality models built on decision trees.

Bayesian models (Gelman et al., 1995) are created by examining the distribution of the single variables or variable groups (V) by each class (O) in the training data as a function of the variable values. Using the concepts of probability theory, we observe the conditional distribution P (V|O). Classification of a data point with an unknown value V = v is obtained by finding the maximum of the following formula for O, based on Bayes' theorem:

$$P\,(O|V = v) = \frac{(P(V = v|O)P(O)}{(P(V = v)}$$

The concept of *artificial neural networks* (Haykin, 1998) originates from the efforts to model the operation of the human brain. We can regard a neuron as a logistic regression model that produces the weighted aggregate signal received through their nerve-endings and activates output nerve-endings by applying the logistic activation function shown in Figure 7. An artificial neural network is formed by connecting nerve-endings to other neurons where certain neurons will receive input data while another group will provide output data according to Figure 10.

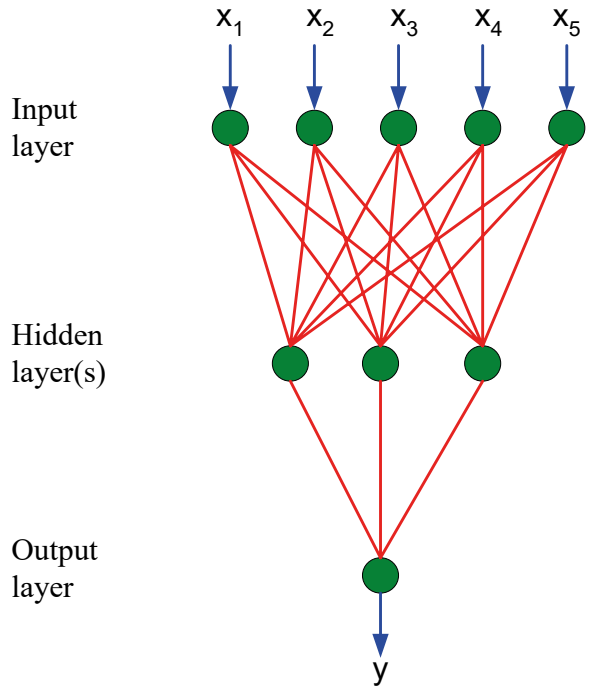

Figure 10
Artificial neural network

Source: The author's own contribution

Artificial neural networks have been used for a long time with moderate success for a wide variety of classification tasks. Today the growth of computing capacities allows for training very large networks (the so-called *deep* networks). Later we will explain the principle of deep learning and the options for its application.

Kernel methods and text mining

The SVM *(support vector machine)* (Vapnik, 1995) method can transform a model that resembles regression and separates data points linearly (see Figure 8) into a model of non-linear classes, such as classes delimited by curves. The key idea behind this model is to transform data to a higher dimensional space by using so-called kernel functions. As shown in Figure 11, a data set that cannot be split in a single dimension can be split into two parts with a line, which can be achieved by applying the kernel function.

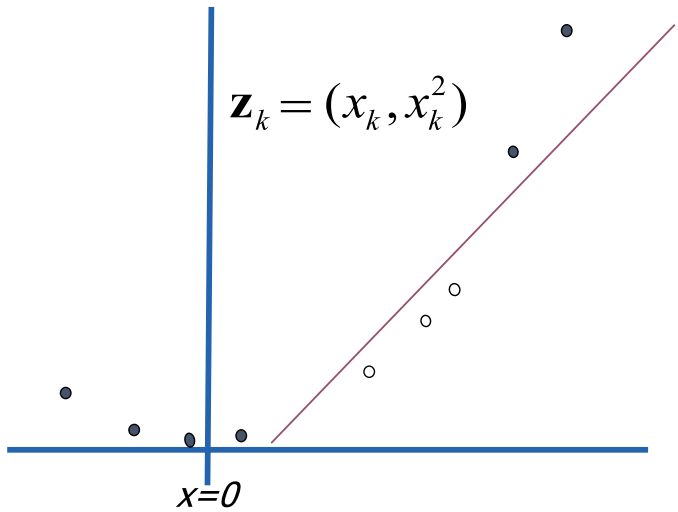

Figure 11

Linear separation by mapping to a higher dimensional space

Source: The author's own contribution

SVM can be used very successfully to classify text data (Joachims, 1998; Tong–Coller, 2001). A proven method of analyzing text is to use the so-called "bag-of-words model", which describes the properties of single documents with the number of words in them or with variables derived from the weighted and transformed versions of those figures (see Figure 4). For bag-of-word data, SVM usually provides high quality classification results. A recent area of application for SVM is the classification of time series. For time series, similarity measures are often difficult to manage mathematically and cannot be directly combined with other variables. The so-called similarity kernel methods, however, can be used to map time series to a space shared with other data types (Daróczy et al., 2015).

Boosting

Boosting is perhaps the most successful and widely usable method of supervised classification (Chen et al., 2016). Its basic principle is the constant correction of the results from very simple classifiers by training more and more simple classifiers that eliminate the errors made by the previous set of classifiers. Its most successful implementations are AdaBoost (Freund–Schapire, 1997), LogitBoost (Friedman et al., 2000) and GBT *(Gradient Boosted Tree)* (Friedman, 2001). Boosting methods typically rely on small-size decision trees that may consist of as few as a single decision or a limited number of levels (two to four) as simple classifiers. A possible instance of the model that corrects the first decision tree with a second tree in the next iteration is shown in Figure 12. An advantage of the model produced by combining two decision trees is that the second tree can be trained by using the entire set of data compared to the single large tree, where the deeper branches may have been determined by using a very small subset of the data points.

```
Tree #1
    if Travels < 1:
        score = -1
    else:
        score = 1
Tree #2
    if Location < 0.4:
        score = score - 1
    else:
        score = score + 1
Tree #3
    if Travels < 2:
        score = score - 1
    else:
        score = score + 1
```

Figure 12

A GBT example using data from Figure 3

Source: The author's own contribution

Deep learning and image classification

Deep learning means training a neural network consisting of a very large number of neurons. Neurons are often interconnected in layers modelling human vision, such as the convolutional network that is capable of recognizing edges in an image or the max pooling network that changes resolution and proceeds from the smaller units towards bigger ones (Figure 13). Deep learning is very successful in recognising images (Krizhevsky et al., 2012) and sounds (Lee et al., 2009) and in resolving special word processing tasks (Ronan–Weston, 2008). Its most recent areas of application includes recommender systems (Wang et al., 2015).

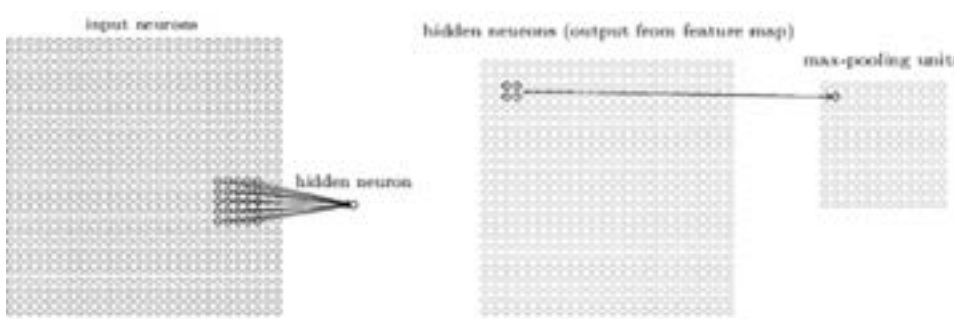

Figure 13

Building blocks of artificial neural networks

On the left: convolutional neuron. On the right: Max-pooling neuron

Source: Teterwak (2015)

Regression

Regression is also part of the supervised learning methodology. Unlike classification, instead of separating positive and negative events, it is numerical values such as risk levels or the extent of damage caused that must be predicted. Most of the classification methods have a regression counterpart, such as the *support vector* regression (Basak et al., 2007). We will explain two basic methods below. The earliest version of the regression method is linear regression that, much like logistic regression (which, contrary to its name, is not a regression but a classification method), produces output values as a weighted aggregate of the variables. Regression trees can be constructed if we assign to the levels of decision trees a continuous numeric value, calculated as the average of data point values associated with that level, instead of a class designation. The boosting method presented in Figure 12 is actually made up of regression trees as we assigned numeric values to the levels.

Clustering

Clustering is a popular tool for unsupervised learning; its similar or synonymous names are segmentation, unsupervised classification, or grouping. This method can be efficiently used for presenting and visualizing small sets of data, and for this reason it is popular in the field of social sciences (Newman, 2001) and in the study of biological systems (Barabasi–Oltvai, 2004). The simplest method of clustering is the k-means algorithm that repeatedly assigns data points to *k* clusters, initially selected at random until they create good separation of the data points (Ball–Hall, 1965). There are a huge number of other known methods, which are summarized, for example, in Jain (2010). In a big data environment, clustering can deliver results that are relatively difficult to interpret and verify as very strongly associated events dominate the entire system. Typically data points are clustered around the centre and the easy-to-interpret groups make up only a negligible part of the data set (Kurucz–Benczúr, 2010) as shown in the examples of Figure 14.

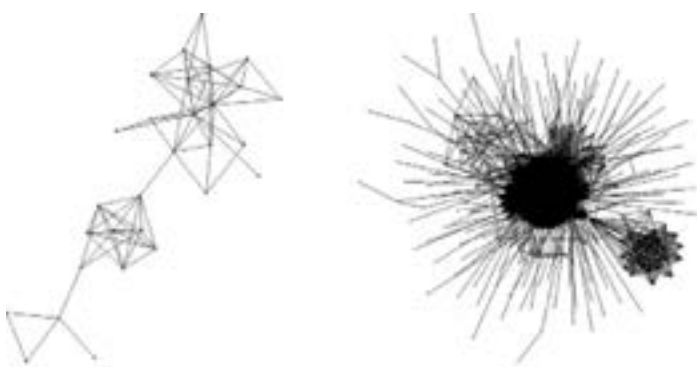

Figure 14

Typical substructures in a very large network: a small number of densely connected vertices that are linked loosely to each other with long appendages.

Source: Subgraphs of Livejournal with 29 and 317 vertices (Kurucz–Benczúr, 2010)

Anomaly detection

Recognition of anomalies (Patcha–Park, 2007) appears to be a very promising method that seems to be very useful for practical applications at first glance. In real cases, however, the fact that recognized anomalies do not necessarily pose a risk is often a problem. If the methods have no access to external knowledge, they can identify as an anomaly all human actions performed only by few people, generating a lot of false alerts. Adding external knowledge often means that an originally unsupervised task is turned into a supervised one, which involves adding labels to known incidents. Two popular methods of detecting anomalies is to check distributions to filter out statistically extreme values and to use classifiers to build a model that fits normal behaviour the best. An example of the latter is the one-class SVM (Manevitz–Yousef, 2001) or the generation and separation of artificial data from normal patterns.

Methods of network science

Network science (Lewis, 2011) is a relatively young field that specializes in the study of the structure, formation, and stability of linked events. In the context of security and risks, persons who are connected with each other, either in real life or virtually on community portals or who are found at the same location and communicate with each other, persons entering into a contractual relationship, or members of organizations and hierarchies can form a network. Visualization of networks will reveal the system of connections between cooperating groups and events, as shown in Figure 15. Network connections can be described using various distance measures. Such terms may include the least number of connections through which a party can contact the other, or the number of common interactions. Complex metrics such as Google PageRank (Brin–Page, 2012) characterizing centrality or SimRank (Jeh–Widom, 2002) measuring similarity are based on the length of random walks within the network.

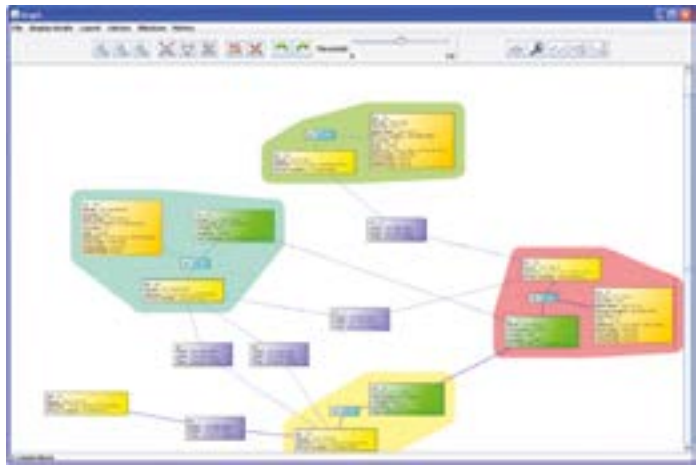

Figure 15
A screenshot of a device providing assistance in the fight against organized crime, displaying network connections

Source: MTA SZTAKI

Challenges of big data

In the past years launching big data projects has become a fashionable trend in all areas of application. This subsection of our study will present examples of the big data phenomenon, and we will attempt to debunk a few misconceptions. Real Big Data tasks are rarely found, Big Data is an exception rather than the rule. Such exceptional cases include web content, social media, and mobile network traffic. Installation of a large number of sensors, for example, in urban or manufacturing plant environments may also generate huge amounts of data, but such sensor networks are not yet widespread. Processing video streams requires special, mostly graphics processing unit (GPU)-based powerful pre-processing and information extraction capabilities and constitute a distinct field of application.

The three "V" – volume, velocity, and variety

Big Data implies tasks that cannot be completed at all with conventional software tools (database management systems, office applications, desktop servers). The most important characteristic of Big Data is that it requires the coordinated computing capacity of a large number of servers, using special software tools. It is surprising that Big Data problems arose while computing capacities were growing. We will use examples to show that data volumes increased at the same pace as the computational capacities. As a result, algorithms that do not scale linearly with the amount of data became more difficult to handle than before. One of the simplest, most basic routine task that does not scale linearly is sorting: it is no wonder that system developers demonstrate the capabilities of their devices by sorting terabytes and then petabytes of data.

Big Data, however, is more than the size of data. Its three key features are as follows:

1. *Volume:* the amount of data, which, in itself, is considered a Big Data sized task if it is in the order of petabytes.
2. *Velocity:* the rate of data influx. Secondary storage devices are often too slow to store the data, which prevents the recording of the entire data content, or the single incidents require too fast a response, for example, in road traffic situation.
3. *Variety:* it greatly increases data processing costs if data is heterogeneous and unstructured, and contains noise or errors. An example is extraction of persons and relations from unstructured text, and recognition of events using camera data.

Quick, cheap, and high quality

In everyday life we usually learn that tasks can seldom be completed quickly, cheaply, and to high standards. Out of the three conditions, two are usually satisfied, but unfortunately the three are almost never. In the case of Big Data systems, the above observation is surprisingly reinforced by a mathematical fact, the famous Fox–Brewer theorem (CAP theorem) (Gilbert–Lynch, 2012). We will describe the theorem and its consequences below.

In the case of IT systems, speed refers to response times, while quality means the accuracy of the received answers. Cheapness corresponds to a much less natural concept called "partitionability". It is very expensive to add large amounts of memory and a high-capacity mass storage device to a single processing server, and after a while costs will rise sharply, thus limiting usability. A cheap solution is that the required computing resources are partitioned over a large number of inexpensive servers that perform distributed computations communicated through the network.

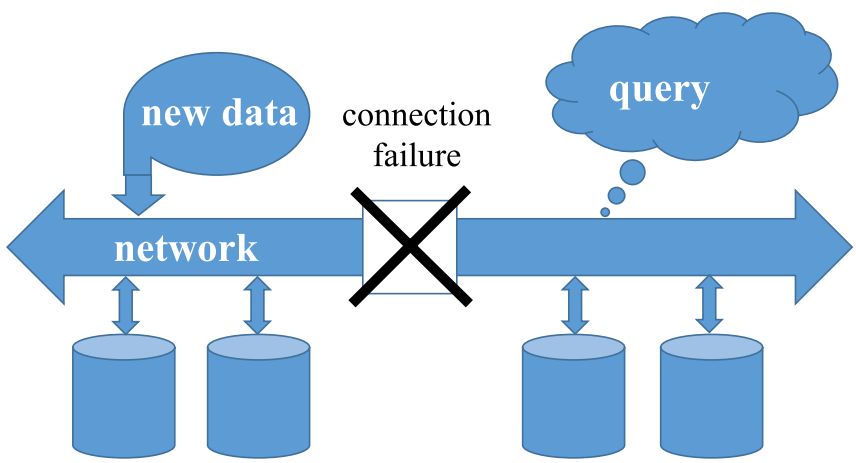

Figure 16

An illustration of the Fox-Brewer theorem

Source: The author's own contribution

The Fox–Brewer theorem states that there is no distributed system (cheapness) that can provide both instant responses (speed) and assuredly correct responses (quality). Below we will describe the very simple proof provided by Gilbert and Lynch (2012). Let us assume that our system is partitioned and responds instantly. In such a case, as shown in Figure 16, it may occur that the system is split into two parts due to a network connection failure. If the left side in the figure receives new data, the right side will not be able to answer even the simplest questions, for example, about the number of records until the network connection is restored.

Let us examine what options are allowed by the Fox–Brewer theorem. We can have a fast and correct solution. Such solutions include conventional single-server systems where data is stored and directly accessed in the memory. The bigger the systems are, the faster the costs of these solutions grow – typically a single server will not be able to accommodate the required amount of memory or occasionally the disk capacity.

We can have a partitioned and correct solution. If the connection to a processing unit is lost, such a system will indicate the failure and attempt to rebuild the data from, for example, redundant server backups. The system will not respond until all failures are eliminated so it will not be fast in every case. We can have a fast and partitioned solution. If a fault or failure occurs, such a system will regard inaccessible data elements as if they did not exist, and will respond to queries without taking those elements into account. The answer will not be correct, but it will at least approximate the correct answer. Once those faults or failures are eliminated, these systems can synchronize missing data and achieve correctness in time.

NoSQL repositories

Database management usually means systems that provide fast and correct answers. Based on what we have so far discussed, we may encounter serious problems when database management systems outgrow servers even with the largest available capacity, and we need to migrate to a distributed system of several smaller servers. A number of open source systems have been released that offer partial support for SQL database queries and are capable of storing data distributed on several servers. Such systems are commonly called NoSQL (Cattel, 2011), which, in the first place, refers to the fact that certain operations are either not supported or only supported by alternate distributed operations. Most often the table joint operation is missing, which would require moving the data of several servers of the distributed data storage. NoSQL is frequently called Not-Only-SQL, but in reality the capabilities of the query language are almost always restricted.

The simplest NoSQL systems are the so-called key-value stores (DeCandia et al., 2007). Next, we will touch upon the most important systems and the principle of *consistent hashing* on which most of these systems are based. The objective of key-value stores is to support the capability of inserting new data by distributing huge volumes of data on several servers, and retrieving them on the basis of keys. The difficulty lies in the fact that in order to prepare for occasional server outage, data should be stored in a redundant way, and we will need to find the server containing the given key even if the primary server is unavailable due to failures.

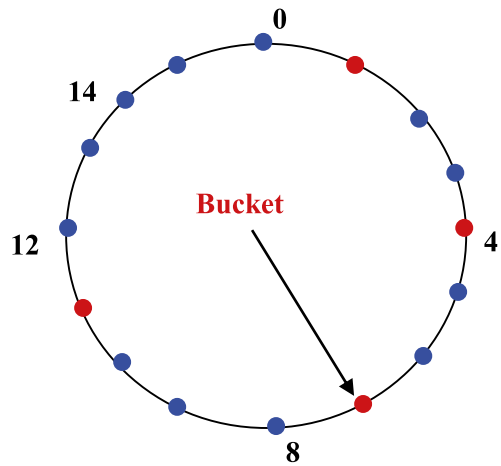

Figure 17
The principle of consistent hashing

Source: The author's own contribution

The basic idea behind key-value stores is the so-called consistent hashing technique (Karger et al., 1997), which is illustrated in Figure 17. Both the keys and the storage servers are mapped to the unit circle. Every key is assigned to some of the closest servers clockwise. In this way we do not need to determine either the number of servers or the number of possible keys in advance. Uniform random mapping ensures that keys are evenly distributed among the servers even if the number of servers changes constantly due to units dropping out of service and replacement servers being started up.

MapReduce paradigm and more

The MapReduce paradigm was introduced by Google (Dean–Ghemawat, 2008), mostly to support building large search indexes. Later they tried to apply it to almost all Big Data solutions, and its open source version called Hadoop (White, 2012) has also been released. MapReduce is a special distributed execution schedule that can be used to implement parallelized tasks on a large number of servers. The general principle is illustrated in Figure 18. The input data of MapReduce – in this case, a big text file – is shown on the left side. In the distributed system, the data is already partitioned across servers. In the first "Map" phase, the elements of the partition are processed one by one, which means counting the number of words in them in this case. Then the result set (word, count) is broken down into pairs and sorted by using "word" as a key. The "Reduce" phase receives values associated with a single key so that the server may count its occurrences in a subset of words.

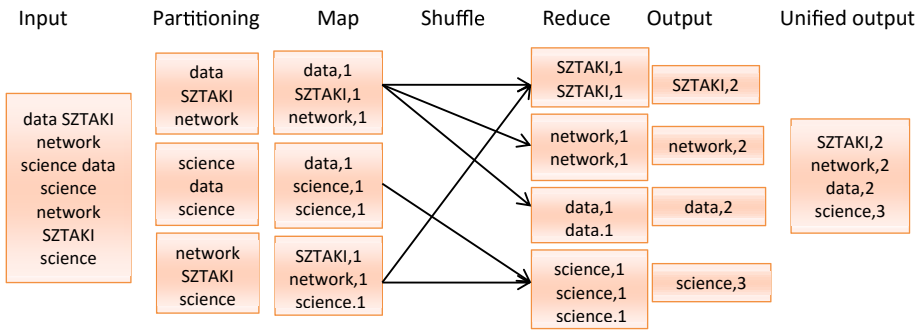

Figure 18

MapReduce example – counting word occurrences

Source: The author's own contribution

MapReduce systems are partitioned and provide correct answers. Correctness is ensured by repeating the individual Map and Reduce steps, possibly after moving them to a server, until successful execution is verified. However, this is exactly their most severe limitation: they are unable to respond quickly. The emergence of the MapReduce paradigm and the open source Hadoop system was a key step towards, among other things, reducing the costs of processing huge volumes of web documents, and making them available to analysts, companies, and authorities. However, MapReduce is no "super weapon," as implementation of complex tasks over big data may be hindered by serious limitations. New approaches requiring a different principle of processing include data stream system, machine learning, and processing of network structures. There are two emerging open source systems that target the above mentioned areas: Apache Spark (Zaharia et al., 2010) and Apache Flink (Carbone et al., 2015).

Summary

This part of our study summarizes applicable data based risk prediction methodologies, the key steps of preparing data and building and evaluating models. Several machine learning techniques offering high quality solutions to supervised learning in the first place were shown. The universal principle of economics – "there is no free lunch" – also emerges in the field of risk predictions: in general, anomaly detection systems that do not require external intervention and perform unsupervised classification do not prove successful in practice. We also emphasized the significance of human labelling, incident investigation and generation of training patterns in machine learning based risk detection. We showed that many data sources available to risk detection fall within the Big Data category, which means that they cannot be processed and analyzed using conventional software solutions. In order to tackle Big Data tasks, we also explored available, mostly open source systems, including their capabilities and limitations.

References

Ball, Geoffrey H. – Hall, David J. (1965): *ISODATA, a novel method of data analysis and pattern classification.* Stanford Research Institute, Menlo Park.

Barabasi, Albert-Laszlo – Oltvai, Zoltan N. (2004): Network biology: understanding the cell's functional organization. *Nature Reviews Genetics,* Vol. 5, No. 2. 101–113.

Basak, Debasish – Pal, Srimanta – Patranabis, Dipak Chandra (2007): Support vector regression. *Neural Information Processing – Letters and Reviews,* Vol. 11, No. 10. 203–224.

Bertino, Elisa – Samarati, Pierangela – Jajodia, Sushil (1997): An extended authorization model for relational databases. *IEEE Transactions on Knowledge and Data Engineering,* Vol. 9, No. 1. 85–101.

Brin, Sergey – Page, Lawrence (2012): Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer Networks,* Vol. 56, No. 18. 3825–3833.

Carbone, Paris – Katsifodimos, Asterios – Ewen, Stephan – Markl, Volker – Haridi, Seif – Tzoumas, Kostas (2015): Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Engineering Bulletin,* Vol. 38, No. 4. 28–38.

Cattell, Rick (2011): Scalable SQL and NoSQL data stores. *Acm Sigmod Record,* Vol. 39, No. 4. 12–27.

Chandola, Varun – Banerjee, Arindam – Kumar, Vipin (2009): Anomaly Detection: A Survey. *ACM Computing Surveys,* Vol. 41, No. 3. Article 15. 1–58.

Chen, Tianqi – Guestrin Carlos (2016): *Xgboost: A Scalable Tree Boosting System*. Source: https://arxiv.org/abs/1603.02754

Cox, David R. (1958): The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society, Series B (Methodological),* Vol. 20, No. 2. 215–242.

Daróczy Bálint – Vaderna Péter – Benczúr András (2015): *Machine Learning Based Session Drop Prediction in LTE Networks and Its SON Aspects*. IEEE 81st Vehicular Technology Conference (VTC Spring). IEEE, Glasgow.

Dean, Jeffrey – Ghemawat, Sanjay (2008): Mapreduce: Simplified Data Processing on Large Clusters. *Communications of the ACM,* Vol. 51, No. 1. 107–113.

Decandia, Giuseppe et al. (2007): Dynamo: Amazon's Highly Available Key-value Store. *ACM SIGOPS Operating Systems Review,* Vol. 41, No. 6. 205–220.

Freund, Yoav – Schapire, Robert E. (1997): A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences,* Vol. 55, No. 1. 119–139.

Friedman, Jerome (2001): Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics,* Vol. 29, No. 5. 1189–1232.

Friedman, Jerome – Hastie, Trevor – Tibshirani, Robert (2000): Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics,* Vol. 28, No. 2. 337–407.

Gelman, Andrew – Carlin, John B. – Stern, Hal S. – Rubin, Donald B. (1995): *Bayesian Data Analysis.* Chapman and Hall, London.

Gilbert, Seth – Lynch, Nancy (2012): Perspectives on the CAP Theorem. *Computer,* Vol. 45, No. 2. 30–36.

Han, Jiawei – Kamber, Micheline (2001): *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Burlington.

Haykin, Simon (1998): *Neural Networks: A Comprehensive Foundation.* Prentice Hall, Upper Saddle River.

Jain, Anil K. (2010): Data Clustering: 50 Years Beyond K-Means. *Pattern Recognition Letters,* Vol. 31, No. 8. 651–666.

Jeh, Glen – Widom, Jennifer (2002): SimRank: a measure of structural-context similarity. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM Press, New York.

Joachims, Thorsten (1998). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.* ECML. Springer, Berlin–Heidelberg.

Karger, David, et al. (1997): Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing.* ACM Press, New York.

Krizhevsky, Alex – Sutskever, Ilya – Hinton, Geoffrey E. (2012): Imagenet Classification with Deep Convolutional Neural Networks. *NIPS 2012,* Pereira, Fernando – Burges, Christopher J. C. – Bottou, Léon – Weinberger, Kilian Q. (eds.), NIPS Foundation, Nevada.

Kurucz Miklós – Benczúr András (2010): Geographically Organized Small Communities and the Hardness of Clustering Social Networks. *Data Mining for Social Network Data,* Vol. 12, 177–199.

Lechun, Yann – Bengio, Yoshua – Hinton, Geoffrey (2015): Deep learning. *Nature,* Vol. 521, No. 7553. 436–444.

Lee, Honglak – Pham, Peter – Largman, Yan – Ng, Andrew Y. (2009): Unsupervised feature learning for audio classification using convolutional deep belief networks. *NIPS 2009,* Bengio, Yoshua – Schuurmans, Dale – Lafferty, John D. – Williams, Christopher K. I. – Culotta, Aron (eds.), NIPS Foundation, Nevada.

Lewis, Ted G. (2011): *Network science: Theory and applications*. John Wiley & Sons, Hoboken.

Manevitz, Larry M. – Yousef, Malik (2001): One-Class SVMs for Document Classification. *Journal of Machine Learning Research,* Vol. 2, 139–154.

MTA SZTAKI: *Jelentés a SCIIMS EU FP7 No. 218223 projektben.* Source: www.sciims.eu/

Newman, Mark E. J. (2001): Clustering and preferential attachment in growing networks. *Physical Review,* Vol. 64, No. 2. 025102.

Pálovics Róbert – Ayala-Gómez, Frederick – Csikota Balázs – Daróczy Bálint – Kocsis Levente – Spadacene, Dominic – Benczúr András A. (2014): RecSys Challenge 2014: an ensemble of binary classifiers and matrix factorization. *Proceedings of the 2014 Recommender Systems Challenge,* ACM Press, New York.

Patcha, Animesh – Park, Jung-Min (2007): An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks,* Vol. 51, No. 12. 3448–3470.

Powers, David M. W. (2011): Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies,* Vol. 2, No. 1. 37–63.

Ronan, Collobert – Weston, Jason (2008): A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th international conference on Machine learning.* ACM Press, New York.

Safavian, S. Rasoul – Landgrebe, David (1998): A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 21, No. 3. 660–674.

Swets, John A. (1996): *Signal detection theory and ROC analysis in psychology and diagnostics: collected papers.* Lawrence Erlbaum Associates, Mahwah.

Tan, Pang-Ning – Steinbach, Michael – Kumar, Vipin (2006): *Introduction to data mining.* Addison-Wesley, Boston.

Teterwak, Piotr (2015): *Learn to Build an App to Find Similar Images using Deep Learning.* PyData, Seattle, 07. 24.

Tong, Simon – Koller, Daphne (2001): Support Vector Machine Active Learning with Applications to Text Classification. *Journal Of Machine Learning Research,* Vol. 2, 45–66.

Vapnik, Vladimir (1995): *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.

Wang, Hao – Wang, Naiyan – Yeung, Dit-Yan (2015): Collaborative deep learning for recommender systems. *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM Press, New York.

White, Tom (2012): *Hadoop: The Definitive Guide*. O'Reilly Media, Sebastopol.

Zaharia, Matei – Chowdhury, Mosharaf – Franklin, Michael J. – Shenker, Scott – Stoica, Ion (2010): *Spark: Cluster Computing with Working Sets.* HotCloud. USENIX Association, Berkeley.