Machine learning methods: An overview

Ravil I Muhamedyev

Institute of Problems of Information and Control, Ministry of Education and Science of the Republic of Kazakhstan. Pushkina 125, Almaty Kazakhstan

High School of management information systems ISMA, Lomonosov st. 1, Riga, Latvia

Corresponding author's e-mail: ravil.muhamedyev@gmail.com

Received 1 December 2015, www.cmnt.lv

Abstract

This review covers the vast field of machine learning (ML), and relates to weak artificial intelligence. It includes the taxonomy of ML algorithms, setup diagram of machine learning methods, the formal statement of ML and some frequently used algorithms (regressive, artificial neural networks, k-NN, SVN, LDAC, DLDA). It describes classification accuracy indicators, the use of "learning curves" for assessment of ML methods and data pre-processing methods, including methods of abnormal values elimination and normalization. It addresses issues of application of ML systems at the processing of big data and the approaches of their solution by methods of parallel computing, mapreduce and modification of gradient descent.

Keywords: machine learning learn ability preprocessing big data map reduce

1 Introduction

The domain of Artificial Intelligence (AI) is quite extensive and includes numerous directions, from logics through methods of text tonality analysis. Traditionally the Strong AI and the Weak Ai are distinguished. The first one is oriented to creation of highly intelligent human-level problem systems, and eventually to creation of thinking machines. Such developments are financed by, DARPA [1], for instance. The book [2] also may be mentioned as one of the first to define the notion itself of the thought and to describe the principles of brain function in a popular way.

Weak AI is oriented to creation of applications that perform this or another intellectual ability of humans or animals. For instance, ability of safe across-country movement, beehive (ant) or distributed intelligence [3], systems of natural selection (referred to as genetic algorithms) etc. Frequently, the field of intelligent agents [4] and multi-agent systems, described in detail in the research papers by Gorodetsky V.I. [5, 6] are also included herein.

In the course of development, AI, being at the cutting-age of scientific research, is gradually changing its matter as a science. If in the beginning of its development, its scope of interests included such problems as bio-identification, text recognition, etc., later on they transformed into the scope of technologies widely used in applied sciences, developments and industries [7]. One of successful directions of artificial intelligence is machine learning (ML). ML is successfully applied to solve of many kinds of problems.

Nowadays ML is an established discipline in many ways. The methodology of application of ML considers selection of relevant data and its pre-processing, selection of adequate algorithms and solution quality assessment. Development of this vast domain includes search for optimal usage of accumulated potential of big and heterogeneous data, search for rapid learning methods and analysis of application features depending on the field of application.

The article describes consistently ML methods that found extensive practical application, methods of data preparation and indicators of classification quality assessment and algorithms. A certain part is dedicated to big data management. The article consists of the following parts.

Part 2 describes taxonomy of ML techniques, formal statement of a ML problem, some popular techniques and methodology of ML application to problems.

Part 3 describes classification quality indicators and methodology of learning curves application in assessment of ML systems.

Part 4 describes methods of data pre-processing.

Part 5 is dedicated to ML techniques application in big data management.

The conclusion briefly resumes the describe techniques and methodologies.

Review Paper

2 Machine learning techniques

Data interpretation is often connected to classification, when a certain object is to be related to one of the previously determined classes, to clustering, when objects are split into initially undetermined groups (clusters) and to forecasting, when by some volume of initial data describing the process background, for example, it is necessary to determine its future state in space or in time. In all the cases, when no strict formal techniques of classification or clustering are used, ML techniques are used extensively.

ML techniques include a vast class of algorithms, starting with solution trees, genetic algorithms, and metric techniques, such as k-NN, SVM, statistical methods, Bayesian networks and ending with artificial neural networks [4, 7, 8, 9]. On the merits this direction is meant to solve the central problem of an intelligent system, anticipatory to all other actions – assessment of a current object (situation).

One of practical applications, where ML techniques are used extensively since 1970's is extraction of commercial minerals. For instance, artificial neural networks are applied in petrography for log data analysis, in lithology for assessment of mineral resources base, in seismic sounding [10-17]. The research paper [18] is dedicated to application of neural networks, the research paper [18] is dedicated to decision of practical interpretation problems of oil production log data. The research papers [19-21] describe some results of feed-forward neural networks application for interpretation of well-log geophysical data during uranium exploration.

Still applications of ML are much more extensive. They include medicine [22-24], biology [25], robotics, municipal facilities and industry [26], service sector, ecology [27], innovative systems of communication [28], astronomy [29] etc.

Herewith the taxonomy of ML, key algorithms and features of their application are being considered.

2.1 TYPES OF MACHINE LEARNING ALGORITHMS

ML, as a part of a vast scientific direction called Artificial Intelligence (AI), on the merits is implementing the potential of AI idea. The main anticipation related to ML considers implementation of demand for flexible, adaptive, learning algorithms or methods of computation [*]. This results into new functions of systems and programs. ML opportunities, i.e. ability of learning and giving expert-level recommendations in a narrow application domain, provide algorithms, which are divided into two big groups, realizing the main property – ability of learning:

- Unsupervised learning (UL)
- Supervised learning (SL)

On top of that, specify

- Reinforcement learning [30] (RL)
- Semi-supervised learning [31] (SSL)

The main problem decided with use of ML algorithms involves relating the observed object to one or another class with the following automatic or manual decision generation. Such problems occur very frequently. As a specific illustration may be given the problems that arise in the process of mobile autonomous robot movement, and relating to recognition of an expected path; recognition problems of face, facial gestures and emotions; analysis of user activity data from e-commerce systems, which allows to perform both interface optimization and planning of system's response. In broader terms, this is a data analysis in different information systems, letting forecast status and classification of objects. Solution approaches of this problem differ.

UL techniques solve the problem of clustering, when the range of initially undetermined objects is split into groups by means of automatic procedure based on the properties of these objects. Whereas the number of groups (clusters) may be given in advance or generated automatically. Such algorithms include the adaptive resonance theory -ART and self-organizing maps- SOM or Kohonen maps [32], and a vast group of clustering algorithms (k-means, mixture models, hierarchical clustering etc.) [33, 34].

SL decide the problem of classification, when finite groups of somewhat determined objects are distinguished in a potentially infinite aggregate of objects. As a rule, groups are formed by an expert. Furthermore the expert can explain or not the reasons of the initial classification.

The algorithm of classification should put the following undetermined objects to this or another expert-organized group by use of the initial classification based on the properties of these objects. SL include a big number of algorithms or a class of algorithms being often divided to linear and non-linear classifiers, depending on the shape (hyperplane or hypersurface) of object dividing classes. In a two-dimension case, linear classifiers divide classes with the only straight line, whereas non-linear classifiers – with the curve (Figure 1).

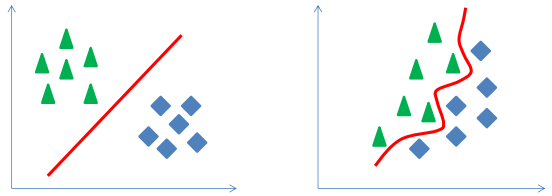

FIGURE 1 Linear (a) and non-linear (b) classifiers

Classification approaches for MO algorithms are presented in research papers [35, 36] in particular. Taxonomy of MO algorithms, not eligible for exhausttiveness, may be presented as the following below hierarchy structure (Figure 2):

Unsupervised learning (UL)
ART
ART1
ART2
ART3
SOM
Generative Topographic Map (GTM)
Cluster algorithms
k-means
K-Means++
K-Medoids
Fuzzy C-Means Clustering Algorithm (FCM)
Soft K-Means Clustering Algorithm (SKM)

[*] Methods of computation is the term that introduced by Donald Knuth to separate mathematically validated algorithms and empirical methods that frequently used in practice

K-Harmonic Means Clustering Algorithm (KHM)
Kernel K-Means Clustering Algorithm (KKM)
Spectral Clustering Algorithm (SCA)
Density models (DM)
Subspace models:
mixture models (MM),
hierarchical clustering (HC)
Supervised learning (SL)
Linear Classifiers
Linear Discriminant Analysis Classifier (LDA)
Logical regression (LR)
Naive bayes Classifier (NBC)
Perceptron (P)
Non-linear Classifiers
Quadratic Classifier (QC)
Diagonal Linear Discriminant Analysis (DLDA)
Support Vector Classification (SVM) (Linear SVM и Non-linear SVM)
Logistic regression (LogR)
k-Nearest-Neighbor (k-NN)
Decision Tree (DT)
Random Forest (RF)
Neural Networks (NN)
Bayesian Networks (BN)
Reinforcement learning (RL)
Q-Learning
Deterministic Q-Learning (DQL)
Monte-Carlo Methods (MCM)
Temporal Difference Methods (TDM)
Sarsa
Semi-supervised learning (SSL)

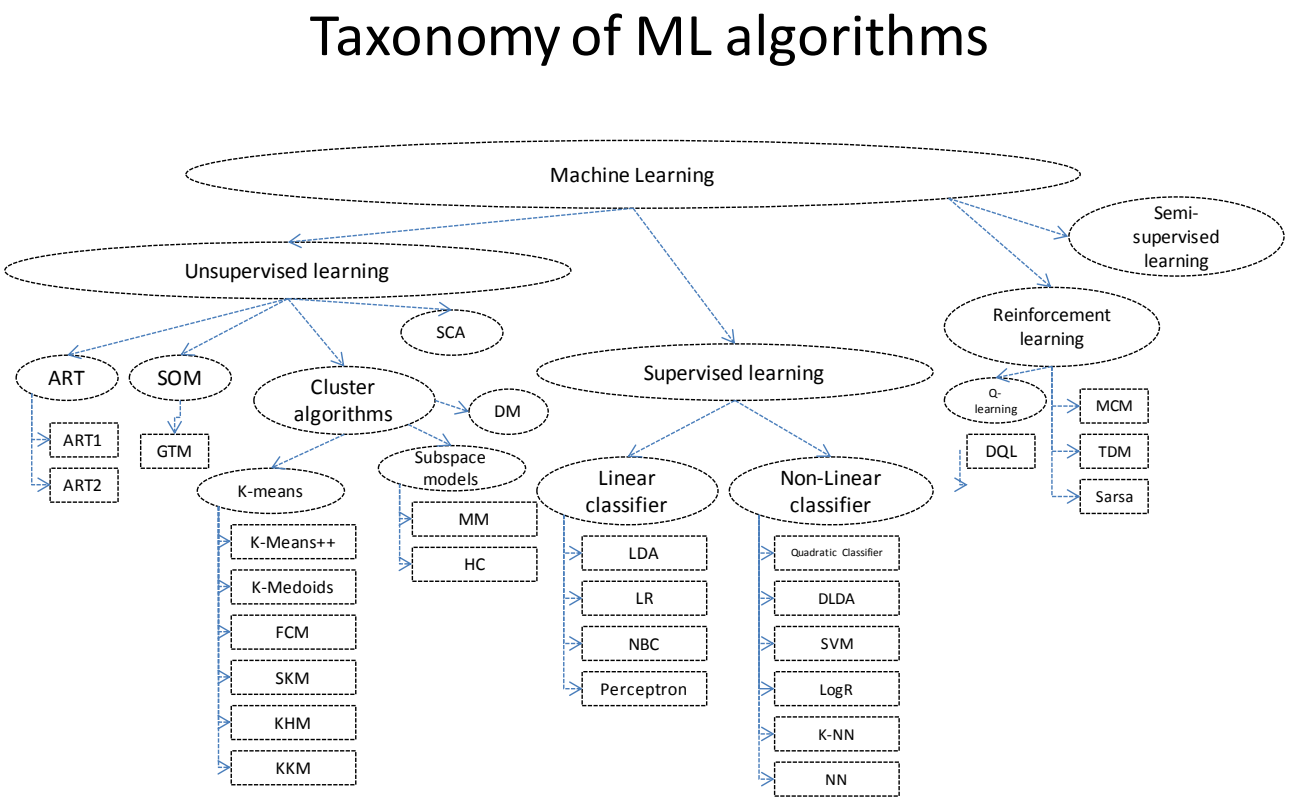

FIGURE 2 Taxonomy of machine learning algorithms

Each of the mentioned algorithms, in fact forms a kind of flock, modified according to these or another needs of programs and algorithms that often differ by computational complexity, difficulty of implementation and learning process automation, ability to classify only two types (binary classification) or several types of objects at once.

Classification and other kind heterogeneous data processsing feature a range of peculiarities:

1) Data heterogeneity that will require introduction of different metrics for certain types of data in some cases
2) Frequent occurrence of big data volumes, when a problem must be solved relatively fast for incoming data portions, which obviously troubles the use of computationally complex learning techniques
3) Complexities of results integration.

2.2 SETUP OF MACHINE LEARNING SYSTEMS

Application of ML techniques in the problems that do not operate any pure mathematical models, only data and probably expert estimates, is often an optimal way of solution. A learning system, an artificial neural network in particular, is able to reproduce the behavior being difficult or impossible to formalize. In supervised learning problems it is often difficult to determine the quality of expert estimates. In a point of fact these problems include also the problems of disease risk detection, product quality assessment, speech recognition, prediction of share quotation level in financial markets, the problem of lithology type recognition, etc. Notwithstanding the experts give a list of relevant features, ranges of measured physical quantities, expert estimates may be contradictory or display errors.

Besides, classification data may contain abnormal values and mistakes, related to physical peculiarities of their

acquisition processes. Consequently a learning system can treat data with mistakes.

Analysis of ML techniques applicability, methods of data processing for application of the mentioned methods, and comparison of algorithms against each other are a necessary condition for development of scientifically proven automatic data interpretation software system.

The general diagram of ML alignment by active problem is presented in Figure 3.

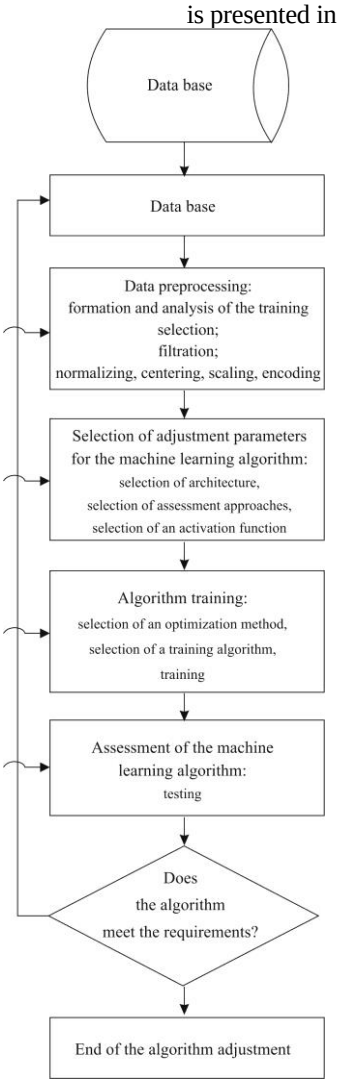

FIGURE 3 Generalized algorithm of a machine learning system alignment by active problem

The alignment technology is an interactive procedure, when correct functioning of the system requires multiple returns to the previous alignment blocks of parameters.

According to this diagram we need to specify the very problem of ML in formal form, to select algorithms to be used in the system, to define quality parameters of ML algorithms, to choose and to validate the methods of data pre-processing.

2.3 FORMAL DEFINITION OF A MACHINE LEARNING PROBLEM

A formal statement of a ML problem (an example-base learning problem or supervised learning problem) is known and to be as follows [37]. Let X and Y be two metric spaces: X (the space of admissible objects), Y (answer or label space) and a (target) function y: $X \to Y$, specified only in the finite aggregate of points (learning selection, sample sets): $y(x^1)$,…, $y(x^m)$, i.e. the labels of objects x^1,…, x^m are known. It requires to build an algorithm A (or "instruct" an algorithm), which calculated a y(x) value by an object x (or „close enough" value, if an inaccurate solution is considered acceptable).

For a finite aggregate $Y = \{1,2,\ldots,l\}$, a problem is called a classification problem (per l non-intersecting classes). In this case it may be considered that a set X is split to classes $K1,\ldots,Kl$, where $Ki = \{x \in X \mid y(x) = i\}$ at $i \in \{1,2,\ldots,l\}$:

$$X = \bigcup_{i=1}^{l} K_i$$

When $Y = \{(\alpha_1,\ldots,\alpha_l) \mid \alpha_1,\ldots,\alpha_l \in \{0,1\}\}$ speak of a classification problem per l intersecting classes. Here *i-class* – $Ki = \{x \in X \mid y(x) = (\alpha_1,\ldots,\alpha l), \alpha_i = 1\}$.

Frequently the loss or cost function $J(A(x), y(x))$ can be used in the problem to describe how "wrong" is our answer $A(x)$ against a true answer $y(x)$. In case of a classification problem it may be considered that.

$$J(A(x), y(x) = \begin{cases} 1, A(x) \neq y(x) \\ 0, A(x) = y(x) \end{cases}$$

But in a regression problem –

$$J\left(A(x), y(x)\right) = |\,A(x) - y(x)|$$

or

$$J\left(A(x), y(x)\right) = (A(x) - y(x))^2 .$$

A problem of learning using a set of training samples presents also an optimization task, which can be decided by search of the minimum value of a cost function $J(\theta)$ across all available examples, defined as the *Sum of Squared* Differences of "a forecasted" value and a real value of y through a wealth of examples *m*. Herein a hypothesis $h_\theta(x)$ "is profiled" to provide the minimum value $J(\theta)$ at a certain set of parameters $\theta_i \in \Theta$.

$$J(\theta) = \min \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 , \quad (1)$$

where m – is an aggregate of samples, h_θ – is a function of hypothese that can be linear $h_\theta = \theta_0 + \theta_1 x$ or non-linear, for instance $h_\theta = \theta_0 + \theta_1 x + \theta_2 x^2$ with a varying set of parameters $\theta_i \in \Theta$.

Running ahead may be said that adjustments of parameters θ_i needs that parameters $x_j \in X$ (in a multi-dimensional case) were expressed in units of the same dimension and approximately of the same value. Most frequently normalization is used to render all the parameters in numbers of the range $0 \le x \le 1$ or $-1 \le x \le 1$. Basically, selection of a normalization function depends on a problem's class. Furthermore, the process of data preprocessing may employ the methods that provide exclusion of anomalous values, exclusion of noises, for example, high-frequency, by the way of fitting, etc. Selection of such methods also depends on a problem class.

After the parameters have been normalized and data is properly framed, there is performed the search of the hypothesis function $h_\theta(x)$, which minimizes the cost $J(\theta)$. For solution of this problem a large number of algorithms is used, partly they are described below.

2.4 REGRESSIVE ALGORITHMS AND DATA CLASSIFICATION ALGORITHMS

2.4.1 Linear regression

A linear regression problem is stated as the search for the minimum cost function (see Formula 1) under the terms that a hypothesis function is a linear one $h_\theta = \theta_0 + \theta_1 x$. Obviously that such function realizes the linear classifier. The gradient descent algorithm is used to obtain an optimal function $h_\theta(x)$, its essence is about the consequent change of parameters θ_0, θ_1 from the following equation

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) , \quad (2)$$

where α is a learning parameter, but $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is a derivative of cost function by θ_j. The sign: = means assignment contrary to the sign of equality (=) in algebraic expressions.

Along with this the algorithm steps are arranged so that initially there occurs the simultaneous change of both parameters based on the Equation (2.2) and only afterwards the assignment of new values to them. Put it differently, the algorithmic sequence of one step of algorithm to be expressed in pseudo code will be the following in case of two parameters

$$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) ;$$

$$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) ;$$

$$\theta_0 := temp0 ;$$

$$\theta_1 := temp1 ;$$

Depending on the learning parameter α, the algorithm may reach the minimum (converge) or, at a too large α, not converge.

The simplest in realization, but not the most optimal in terms of time complexity the "Batch" Gradient Descent algorithm uses all learning examples through each step of the algorithm. In search for parameters θ instead of the gradient descent algorithm the matrix equation can be used $\Theta = (X^T X)^{-1} X^T y$,
where Θ is a vector of parameters, $(X^T X)^{-1}$ is a inverse matrix of $X^T X$, X^T is a transpose matrix of X.

The advantage of matrix operations is that there is not necessary to try the parameter α and to iterate several times. The disadvantage refers to the necessity of getting the inverse matrix, which computational complexity is proportional to $O(n^3)$, and to impossibility of getting the inverse matrix in several cases.

2.4.2 Polynomial regression

Contrary to linear regression polynomial one operates non-linear function of the hypothesis $h_\theta = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + ... + \theta_n x^n$, which allows to plot extrapolated curves (hypersurfaces) of complex shape, but increases the number of parameters and computational complexity. Apart from that, the danger of "relearning" exists, when a curve becomes too complex it fits well the learning set, but it produces a large error across the test set.

In the case when a classifier loses its ability to generalize regularization is used to decrease the effect of high order values

$$J(\theta) = \min \frac{1}{2m} [\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2]$$

Growth of the parameter λ results into extension of the ability to generalize the algorithm. The function of hypothesis turns into the straight line within limits at a very large value of the parameter.

2.4.3 Logistic regression

It is used if objects have to be divided into two classes, for instance, to "negative" and "positive". In this case the function of hypothesis requires fulfillment of the condition $0 \le h_\theta(x) \le 1$, which is achieved with use of a sigmoid (logistic) function

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}},$$

where Θ - a vector of parameters

Can be expressed also as

$$h_\theta(x) = g(\Theta^T x),$$

where $g(z)$ - a sigmoid function.

Let us remark that a sigmoid function also is widely used in neural networks as an activation function of neurons, as it is continuously differential and hence guarantees convergence of algorithms of neural network learning. The sample of a sigmoid is demonstrated in Figure 4.

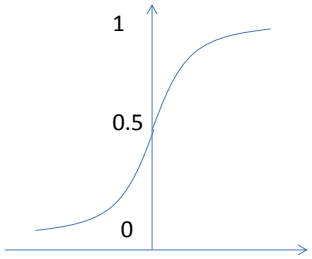

FIGURE 4 Sigmoid function

$h_\theta(x)$ can be considered as the likelihood that the object can be "positive" $h_\theta(x) \ge 0.5$ or "negative" $h_\theta(x)$ <0.5. In complex cases that requires a non-linear interface, for example in the shape of a circle (Figure 5) - $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$

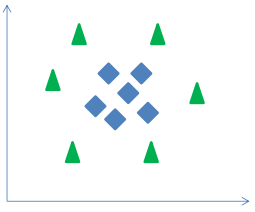

FIGURE 5 Non-linear interface between objects of different classes

Adjustment of parameters Θ, after selection of the function of hypothesis, is executed with regard to minimize the cost function as follows

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))]$$

Same as in case of a linear regression, minimization is reached with use of the gradient descent algorithms, though Conjugate gradient [38], BFGS, L-BFGS [39] is also applicable.

Logistics classifier can be used also for the case of several classes. Here, the classifier is adjusted separately for each class. The class, which a new object belongs to, is computed by value calculation of all functions of hypothesis and selection of maximum rated value among them $\max_i h_\theta^{(i)}(x)$, where i is a class number.

Put it differently, the object belongs to the class with the maximum function of hypothesis.

Same as in case of a linear regression, regularization is used to increase the generalizing ability of the algorithm

$$J(\theta) = [-\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

2.4.4 Artificial neural networks

Artificial neural networks (ANN) is an apparatus being actively researched since 1940's. NN, as a part of the theory of connectionism, has passed the prominent path from the epoch of overestimated expectations, afterwards, through the epoch of disappointments in 1970's, to a widely applicable technology at present. The link between biological neurons and opportunities of their modeling with use of logistic computations is stated in the paper Warren S. McCulloch, Walter Pitts [40], in the Rosenblatt's paper [41] the model of perceptron is described, in the books of Minsky M. and Papert S. [42-44] limitations of a single-layer perceptron are elicited. In 1974 Paul Werbos suggested the algorithms of back propagation [45, 46] applicable for learning of a multilayer perceptron or a neural network. The most popular and helpful ANN is a network of forward propagation, where non-linear elements (neurons) are represented as consequent layers, but information is distributed in one direction [47] (feed-forward neural networks). In 1989 the research papers by Cybenco G. [48] and Hornik K. etc. [49] demonstrated that such a network is able to approximate practically any function.

The theory of connectivism was contributed considerably by national scientists [50-53], who demonstrated the possibility to decide classical computational problems based on neural networks, thus putting a fundamental principle of neural computers generation.

Application of a neural network apparatus addresses decision of a wide range of computationally complex problems, such as optimization, signal processing, image recognition, forecasting and classification.

Let us consider application of feed-forward neural networks. An individual neuron represents a logistic element, consisting of input elements, an integrator, an activating elements and a single output (Figure 6).

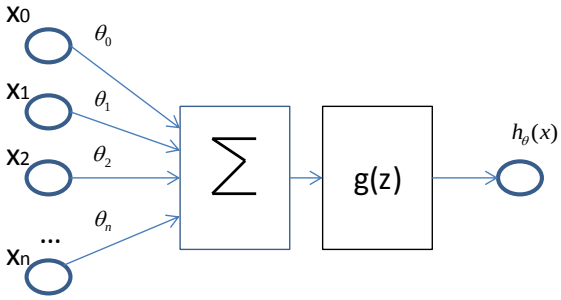

FIGURE 6 Diagram of a classical neuron

Neural output is determined with the formulas

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \ + ... + \theta_n x_n$$

$$h_\theta(x) = g(z),$$

where $g(z)$ – a sigmoid function.

To make the diagram simpler the integrator and the activating element are united, thus a multilayer network can look as in the Figure 7. The network contains three input neurons, three hidden layer neurons and one output neuron. In the Figure input neurons are marked with the symbol x, hidden layer neurons – with the symbols $a_1^{(2)}, a_2^{(2)}, a_3^{(2)}, a_0^{(2)}$, and output layer neurons – with the symbol $a_1^{(3)}$.

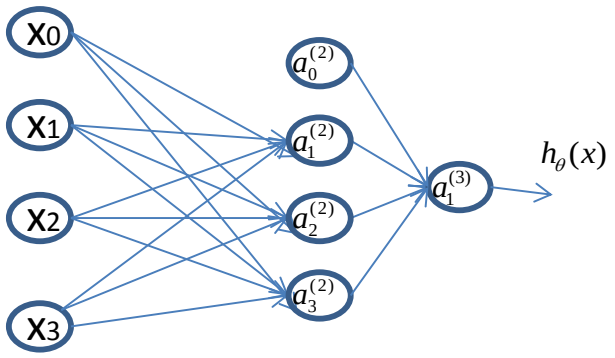

FIGURE 7 Diagram of a multilayer network

Output of each hidden layer neuron can be calculated same as the one for a single neuron:

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3)$$

The neural network output is determined from the equation:

$$h_\theta(x) = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$$

The benefit of the neural network is the opportunity to classify several classes at once. In this case the output layer contains the number of neurons equal to the number of classes. For example, as may be required to classify objects of two classes we will get a network (Figure 8).

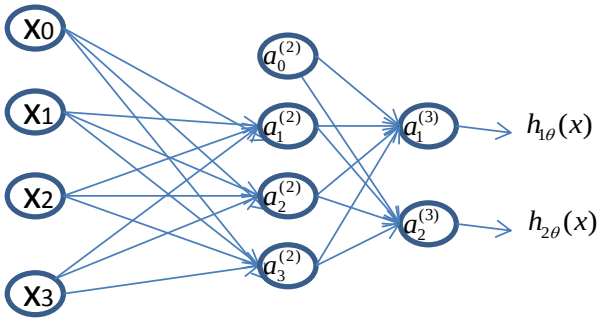

FIGURE 8 Diagram of a multilayer network with double output

To adjust the weights θ of the neural network (training of the network) they use the cost function, reminding the cost function for logistic regression.

$$J(\Theta) = -\frac{1}{m}[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - h_\Theta(x^{(i)}))_k] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}(\Theta_{ji}^{l})^2,$$

where L –number of neural network layers,

S_l - number of neurons in the layer l,

K – number of classes (equal to the number of neurons in the output layer),

Θ - a weight matrix.

To train a multilayer neural network they use the back propagation error algorithm and different modifications targeting to accelerate the training process.

2.4.5 k-Nearest-Neighbor (k-NN) [54, 55] algorithm

The algorithm is based on calculation of the number of objects in each class of the sphere (hypersphere) with the centre in the recognized object. The object belongs to the class, which objects dominate in this sphere. This technique supposes that weights have been chosen individually for every object.

If weights are not same, instead of calculation of the number of objects their weights can be added together. Thus, if the sphere around the recognized object contains 10 reference objects of class A with the weight 2 and 15 error/boundary objects of Class B with weights 1, the point will be referred to Class A.

Weights of objects in the sphere can be expressed as inversely proportional to their distance to the recognized object. Thus, the closer is an object, the more significant it is for this recognized object.

In total, a metric classifier can be described as:

$$a(u; X^l) = \arg\max_{y \in Y} \sum_{i=1}^{l} [y_u^{(i)} = y] w(i,u),$$

where w (i, u) – a weight of an i-neighbour of an recognized object u, a (u;X^l) – class of an object u, recognized by a selection X^l.

The radius of the hypersphere can be as constant, as dynamic. Moreover, in case of a dynamic radius, a radius of each point is adjusted so that the number of objects in each sphere is constant. Thus, for recognition in areas with varying density of selection, the number of „neighboring" objects (being actually used in recognition) will be the same. In this manner, there is excluded the situation when recognition suffers scarcity of data in low-density areas.

2.4.6 Support Vector Classification (Linear SVM and Nonlinear SVM) [56]

This algorithms refers to the group of boundary techniques: it determines classes with use of area boundaries. The technique is based on the concept of solution planes. A solution plane divides objects of different classes. In high-dimension spaces instead of straight lines there should be considered hyperplanes of one dimension less than the one of the assumed space. In R3, for example, a hyperplane is a two-dimensional plane.

The support vector technique searches for samples, located on class boundaries (at least two), i.e. support vectors, and decides the problem of finding the division a set of objects to classes with use of a linear decision function. The support vector technique builds classification function f(x) as follows:

$$f(x) = sign(\langle w, s \rangle + b),$$

where < , > – scalar product,

w – normal (perpendicular) vector to a dividing hyperplane,

b – auxilliary parameter equal to the absolute module of a distance from a hyperplane to the origin of coordinates. If a parameter b equals zero, a hyperplane passes through the origin of coordinates.

Objects with f(x)=1 get into the first class, but objects with f(x)=-1 - into another one.

From the point of classification accuracy the best straight line to be selected is the one maximally distant from each class. Such straight line (in general — a hyperplane) is identified as an optimal dividing hyperplane. The problem consists in selection of w and b, maximizing this distance.

In case of non-linear division the approach of support vector machine adaptation is available. It is necessary to input the space of Rn indicators into the space H of a larger dimensionality using the function of the form: φ = Rn → H.

Hence decision of the problem reduces to a linear divisible case, i.e. a dividing classification function is searched again as: $f(x) = sign(\langle w, \phi(x) \rangle + b)$.

Also possible another option of data conversion – conversion to polar coordinates:

$$\begin{cases} x_1 = r\cos(\phi), \\ x_2 = r\sin(\phi). \end{cases}$$

2.4.7 Linear Discriminant Analysis Classifier (LDAC) [57]

In mathematical statistics classification is often identified as discrimination. Discriminatory analysis is the section of multivariate statistical analysis, which includes statistical classification methods of multivariate observations.

Linear discriminatory analysis is a statistical and ML method used in search for linear combination of variables eminently suitable for division of two and more classes of objects or events. LDA tries to express a dependant variable (class label) via a linear combination of other indicators or measurements. The acquired combination can be used as a linear classifier. The indicators used to distinguish one class (subset) from another are often identified as discriminatory variables.

Let the learning selection be expressed in matrixes X_1 and X_2, having by I_1 and I_2 lines (objects). The number of columns (variables) is the same. Initial assumptions are as follows:

- Each class (k=1 or 2) represented with normal distribution;
- Covariance matrixes of these two classes are equal $\Sigma_1 = \Sigma_2 = \Sigma$.

The rule of LDA classification is the following: a new sample x is referred to that class, which it is closer to in *Mahalanobis* units:

$$d_k = (x - \mu_k)\sum^{-1}(x - \mu_k)^t, k = 1, 2$$

In practice unknown mathematical expectations and covariance matrix are replaced with their estimates

$$m_k = \frac{1}{I_k}\sum_{i=1}^{I_k} x_i ,$$

$$S = \frac{1}{I_1 + I_2 - 2}(\tilde{X}_1^t \tilde{X}_1 + \tilde{X}_2^t \tilde{X}_2),$$

where $\tilde{X}_k$ – a centered matrix X_k.

If we set equal a distance d1 to a distance d2, d1=d2, in the mentioned above forula, it makes possible to find the formula of a class division curve. Along with this quadratic terms are reduced and the equation becomes linear

$$xw_1^t - v_1 = xw_2^t - v_2,$$

where

$$w_k = m_k S^{-1},$$

$$v_k = 0.5 m_k S^{-1} m_k^t .$$

The values in different parts of the equation are identified as LDA-accounts, f1 and f2. The sample refers to class 1, if f1 > f2, and, vice versa, to class 2, if f1 < f2.

2.4.8 Diagonal Linear Discriminant Analysis (DLDA)

Algorithm is a kind of the Linear Discriminatory analysis, described above. This algorithm was described in 2002 [58] as an optimized LDA modification for multivariate data (high-dimension problems). When all class densities have same diagonal covariance matrixes:

$$\Delta = diag(\sigma_1^2 \ldots \sigma_G^2).$$

The classification rule will look as follows:

$$d_k = \sum_{i=1}^{G} \frac{(x_i - \mu_{ki})^2}{\sigma_i^2} .$$

3 Quality assessment of machine learning systems

In a view of abundance of algorithms selection of their use in a certain ML problem can become complicated. To compare algorithms and methods among themselves and probably with an expertise results they use indicators of algorithm classification accuracy and learning curves. The function of accuracy indicators is to give the estimate demonstrating how much the certain ML classifications or forecasts differ from the ones made by experts. Along with this frequently use the elementary indicator – percentage of correctly classified examples. For estimation of type I and II errors also use some important indicators: "precision", "recall" and summarizing indicators - T1 Score and kappa. Their use is especially important in case of unequal by volume classes, whereas the number of objects of one type significantly exceeds the number of objects of another type. One more important indicator of ML method is its learning ability, i.e.

improving its accuracy indicators as the number of samples is increasing. At this point, it might happen that the method showing very good results with train set of examples will produce unsatisfactory results with a testing one, i.e. it does not feature the necessary degree of summarizing. The balance between summarizing ability and accuracy can be found with help of "learning curves", which in a general case are able to show if this or another method can improve its result in a way that its accuracy indicators were approximately equal and satisfied the research domain requirements as for training as for testing setoff values.

3.1 INDICATORS OF CLASSIFICATION ACCURACY ESTIMATES

At present in the field of ML quality assessment most frequently apply:

Accuracy – fraction of correctly classified examples (percentage of correctly classified examples)

$$Ac = \frac{N_t}{N},$$

where N_t - number of correctly classified examples, N – total number of objects.

This indicator is essentially important, but if the number of objects in classes is substantially unequal, said to be uneven or "skewed" classes, it might happen that a very bad classifier would produce a large value Ac. For example, if type I objects make 90% of total number of objects, but type II objects only 10%, it will be sufficient for a classifier just to report on recognition of a type I object and its accuracy will reach 90%. Thus, even if the algorithm will never recognize correctly a type II object, still it will have a high indicator Ac. At this, if recognition of type II objects is of prime importance, an indicator Ac will just mislead. To avoid such an inadequate estimate some more important indicators are taken: "precision", "recall" and a summarizing indicator - T1 Score, calculated from the following equations:

Precision: $P = \frac{T_p}{(T_p + F_p)}$,

Recall: $R = \frac{T_p}{(T_p + F_n)}$

T1 Score: $T1Score = \frac{2PR}{(P + R)}$

Let us explain the given equations.

Let us concern the case of classification of two classes (or one type I class and all other classes, which we assign the number 0) to. In this case the following situations in Table 1 are optional.

TABLE 1 Classification of two classes

		Actual class	
		1	0
Predicted class	1	**True positive**	False positive
	0	False negative	**True negative**

Cases True positive (Tp) and True negative are the cases of correct operation of a classifier, appropriately, cases False negative (Fn) and False positive (Fp) of incorrect operation. At this, Fn can be concerned as the sign of an excessively pessimistic (cautious) classifier, Fp – vice versa, as the sign of excessively optimistic or incautious classifier. Then

Precision: $P = \frac{T_p}{(T_p + F_p)}$

will show a part of correctly recognized objects of the specified class referring to the total number of objects taken by the classifier as objects of the specified class. Alternatively, **recall**

$$R = \frac{T_p}{(T_p + F_n)}$$

will show the ratio of correctly recognized objects to the total number of objects of the given class.

Both indicators and P and R show "confusion" of a classifier. Still P shows how optimistic is a classifier in its estimates, or how frequently it "likes" (low value of P) adding objects of other classes to the given one. While R shows how "pessimistic" is a classifier in its estimates, i.e. how frequently it neglects (low value of R) objects of the needed class.

Evidently, it is desirable that both of these indicators tend to 1. For some "average" estimate use

$$T1Score = \frac{2PR}{(P + R)},$$

which, as is clear from the equation, also tend to 1, if both indicators P and R are close to 1.

Let us notice that use of simple Average= (P+R)/2 can result into a wrong idea about properties of algorithm. For example, let us assume three algorithms showing the following Precision и Recall estimated (Table 2)

It is evident that a simple average (column Average) gives the highest estimate of an absolutely bad algorithm 3, which take all objects as required in error (P are very few). At the same time T1 Score shows the more correct result, giving the highest score to the algorithm 1, which shows close Precision and Recall estimates, and, consequently, will be weighed in their estimates.

TABLE 2

	Precision (P)	**Recall (R)**	**Ave-rage**	**T1 Score**
Algorithm 1	0.55	0.44	0.495	0.4888889
Algorithm 2	0.71	0.12	0.415	0.2053012
Algorithm 3	0.03	1	0.515	0.0582524

For a finer estimate of developed algorithms also use indicators of errors designed for parts of a selection: error on a control selection, error of cross-validation and validation techniques: fold validation, random subselection (subsampling) validation.

Moreover, for the problems with greatly differing numbers of class representatives, the cost function (errors) can be computed in a specific way, for example, in case of skewed classes:

$$J(\theta) = \min \sum_{k=1}^{2} \frac{1}{|\{t \mid y(x_t) = k\}|} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

3.2 "LEARN ABILITY" OF ALGORITHMS

The Estimation of recognition algorithms by simple comparing the quality indexes (accuracy, precision, recall) has the disadvantage that makes it impossible to evaluate the algorithms in the dynamics of change in the volume of training sample. Particularly, in terms of neural networks, accuracy indicators are effected substantially by the number of hidden layers and the number of training examples. With the use of linear regression its order, for k-NN – is the nearest neighbours circle radius, etc. Along with this it is important to consider the algorithm's ability of learning, overfitting or underfitting. The correct balance between underfit and overfit means the search of an algorithm and its parameters, which would be able to show consistent results for a testing set (or a cross validation set). An underfit algorithm will show equally inconsistent results both for test and train sets, while an overfit algorithm will demonstrate a high result for a train set and a low one for a testing set. Let us assume, in case of regression, the appropriate formulas of the curves, extrapolating distribution of training examples as shown below:

a) $\Theta_0 + \Theta_1 x$ - high bias(underfit)

b) $\Theta_0 + \Theta_1 x + \Theta_2 x^2$ - just right

c) $\Theta_0 + \Theta_1 x + \Theta_2 x^2 + \Theta_3 x^3 + \Theta_4 x^4$ - high variance (overfit)

Extrapolation results at some hypothetic distribution of train set objects are shown in Figure 9).

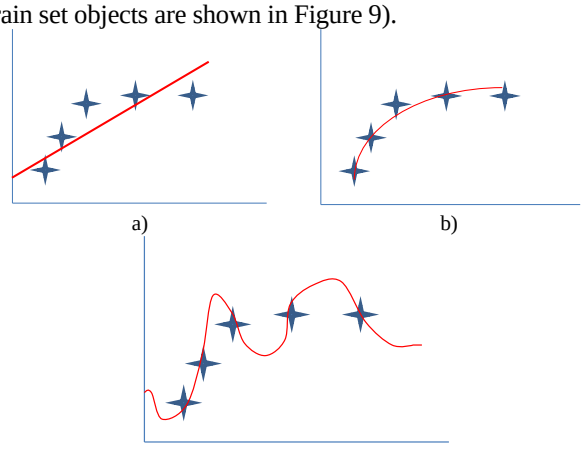

FIGURE 9 Chart of ML algorithm underfit and overfit
a) too linear divisor (underfit) b) almost ideal case
c) too many variables (overfit)

At this error indicators across train and test (cross validation – cv) sets are determined from identical equations (changes only a set of examples)

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 ,$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2 ,$$

where m – train set of examples, m_{cv} - test set of examples (cross validation - cv),

h_θ – function of the hypothesis, which can be linear $h_\theta = \theta_0 + \theta_1 x$ or non-linear, for example, $h_\theta = \theta_0 + \theta_1 x + \theta_2 x^2$ with different set of parameters $\theta_i \in \Theta$.

Model parameters that define the function $h(\Theta)$, are calculated with use of a train set, and validated by examples from a testing set.

Looking ahead, it might be stated that compensation of excess variables in case of overfit the regression model applies regularization, achieving that variables of higher order showed lower effect. The cost estimate formula with consideration of regularization is specified below.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 ,$$

where λ - a regularization parameter.

If a neural network is used reduction in the number of hidden layers of a neural network performs the similar function.

Application of regularization or reduction of the number of hidden layers increase summarizing ability of a ML algorithms and consequently decreases learning ability, in terms of flexibility of adjustment according to subtle differences between classes.

Let us notify that ML systems can be classified as high bias, having a comparatively low capability to generate complex interpolatory curves, and as high variance systems, being able to form curves (surfaces) of complex shape. Pattern of these algorithms (models) differs when the number of training examples increases. Te first ones, as a rule, summarize results ignoring frequently some, probably, essential differences between examples. The second ones, on the contrary, "trace" all nuances, probably random, but at the same summarize insufficiently. The first ones feature underfit, but the second ones – overfit (Figure 2.5) As a rule, it is impossible to evaluate model abilities during one experiment, as both first and second ones can produce close error indicators.

In this context, trainability of ML algorithms is estimated with help of so called learning curves, taking into account the following empirical patterns:

- In normal environment, with an effective algorithm, with increasing the number of training examples the error across the train set will slightly grow, but the error across the test set will decrease (Figure 10);
- If a system is comparatively linear (high bias) increase in the number of training examples will be of little use. The error both across the training and test sets will be approximately equal and large (Figure 11);
- In case of a high variance system, increase in the number of training examples will lead to reduction of the error value across the test set, but it will differ substantially from the error across the train set (Figure 12);

To decrease the test set error even more it is possible to increase essentially the training set (which is not always possible).

Thus, to estimate which of two groups the analyzed algorithm belongs to (too linear or too flexible) it is recommended to analyze the curve of learning errors with increasing body of the learning set, for example, if curves show convergence, but in parallel a high level of errors, this may point to a linear model (impossible to train it).

If unwilling properties of the algorithm have been detected it is possible to try to adjust it, to change the body of the learning set of examples or to choose additional properties of objects, by taking into account the following:

- increasing the size of the train set is helpful at high variance of the algorithms (many layers of neural network, high order of regression), when the program does not feature the needed degree of summarizing, and is adjusted mainly to the train set of examples and cannot normally classify examples from the test set (overfit error).
- Decrease in number of used properties or parameters is helpful at high variance of the algorithms (many layers of the neural network, high order of regression), i.e. once again for the cases of overfit, and when in parallel the number of training examples is impossible to be increased substantially;
- Use of additional properties is helpful in case of too linear algorithms (low order of regression, few neurons in hidden network layers, or few hidden layers), when the program will show the same bad results both on the test and train sets (underfit errors);
- Use of special synthesized (polynomial) properties, showing higher degrees and products of basic (x_1^2 , x_2^2 ,x_1 x_2 ,…) is also helpful in case of too linear algorithms (low order of regression, few layers of the neural network) (underfit model).

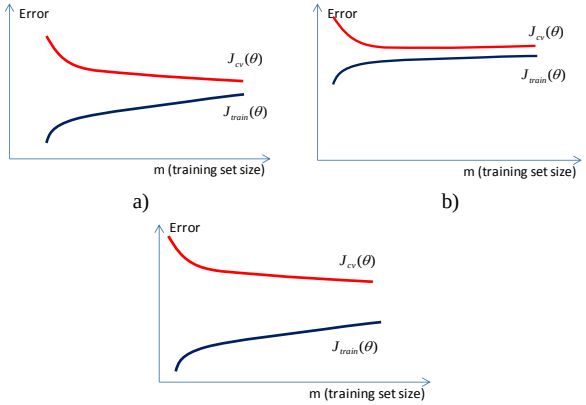

FIGURE 10 Learning curves
a) normal situation, learning algorithm with good summarizing ability
b) hard-trained algorithm, a high bias system
c) high variance algorithm

4 Data preprocessing

ML method use require to render the processed data to a certain format that would allow to feed them to learning and analysis algorithms input. Along with this important cycles of processing are performed, including as a the rule, cleaning of abnormal values; data normalizing; "fitting"; data reformatting; formatting of input data set, which for example can include formatting of a so-called dockable pane, needed for analysis of patterns of presented data sequences; data reconciliation, for example when one data set is time, distance, spectrum, etc. translated, with respect to others, etc. Let us notice that if the set of ML algorithms is known and the properties of these algorithms are mostly well studied, the processes of data preprocessing vary and depend directly on the object domain and quality of available data. Frequently, the specified above "classical" set is added with additional stages, letting finally raise the quality of data interpretation.

4.1 ELIMINATION OF ABNORMAL VALUES

4.1.1 Normal distribution-based algorithm of abnormal value detection

The normal distribution-based algorithm for abnormal activity detection is designed on the assumption that the whole set of "correct" objects forms Gaussian distribution (normal distribution), i.e. *x* values are distributed according to the normal law of distribution, defined by mathematical expectation - μ and mean square deviation - δ^2

$$x \sim N(\mu, \delta^2),$$

which can be presented graphically with Figure 11.

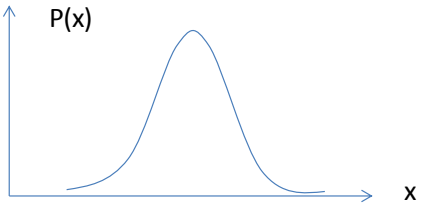

FIGURE 11 Approximate curve of x objects probability distribution subordinated to normal (Gaussian) law

Along with this probability of that or another value is calculated from the known formula

$$p(x;\mu,\delta^2) = \frac{1}{\sqrt{2\pi\delta}} \exp(-\frac{(x-\mu)^2}{2\delta^2}).$$

If objects have several properties, then probabilities will subordinate to the multivariate Gaussian distribution.

At this, $p(x) = \prod_{j=1}^{n} p(x_j;\mu_j,\delta_j^2)$, where n – number of object properties to be classified (number of parameters).

In this manner, the calculation algorithm of abnormal objects is as follows:

Step 1. Based on m train set examples the parameters of the multivariate Gaussian distribution are determined - $\{\mu_1,...,\mu_n;\delta_1^2,...,\delta_n^2\}$:

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)},$$

$$\delta^2{}_j = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2 ,$$

where $x_j^{(i)}$ - a j-parameter of an x object in the training example I from the set of examples m

Step 2. For each new x item its probability is calculated

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \delta_j{}^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\delta}_j} \exp(-\frac{(x_j - \mu_j)^2}{2\delta_j{}^2}) .$$

Step 3. If the calculated probability is lower than some threshold values ε , then this object x is considered abnormal

If $p(x) < \varepsilon$ then x is abnormal.

In practice, training a system to detect abnormal values considers detection of the threshold value ε . Adjustment of this threshold can be performed based on available examples (similarly to supervised algorithms). After the series of experiments such a threshold ε is established so that all (or most part of) "wrong" objects would be detected as abnormal.

The applied approach of abnormal value detection is illustrated graphically with Figure 12

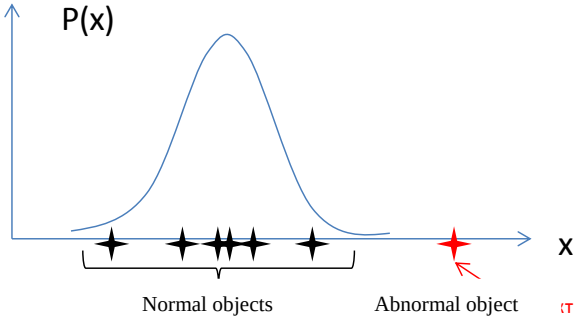

FIGURE 12 Exclusion of abnormal objects

Let us notice that if the distribution differs from normal, it can be normalized by calculating the logarithm of *x* or the degree of *x* values.

4.1.2 Use of ML for detection of abnormal values

Obvious, that supervised learning algorithms are quite feasible for detection of abnormal objects. In this case abnormal objects represent an individual class of objects, which can be detected with algorithms k-NN or logistical regression. In both cases adjustment of those object properties being essential in segregation of abnormal values.

4.2 DATA NORMALIZING AND CENTERING METHODS

4.2.1 Linear normalizing

In a view of cleaning the abnormalities data and operation comfort perform data normalizing and centering so that each component of the input vector is located on the segment from 0 to 1 or from -1 to 1. If knowing the change range of input variable there can be used the simplest type of conversion

$$p = \frac{(x - x_{\min})(b - a)}{(x_{\max} - x_{\min})} + a ,$$

where [a,b] – the range of acceptable input signals; $[x_{\min}, x_{\max}]$ – the change range of input variable values; p – conversed input signal.

When using this method of conversion (linear normalizing) big change range of input variable can become a problem. In such cases it is possible to use conversion from equations of the sigmoid function or the hyperbolic tangent.

When coding quantitative variable in a general case it is necessary to consider the profound characteristic value, its location over the interval of values, accuracy of measurement. Conversion can be performed using two equations:

$$x_i = \frac{x_i - M(x_i)}{q(x_i)}$$

or

$$x_i = \frac{x_i - M(x_i)}{\max|x_i - M(x_i)|} ,$$

where x_i - an i-coordinate of an input vector X;

Sampling estimate of mathematical expectation x_i (mean value):

$$M(x_i) = 1/n\sum_{i=1}^{n} x_i$$

Sampling estimate of the mean square deviation:

$$q(x_i) = \sqrt{(1/n\sum_{i=1}^{n}(x_i - M(x_i))^2}$$

To make minor changes of big values significant (for example, when the value of an input variable can reach 10000, but in terms of analysis significant is the value change by 1), three kinds of numerical data preprocessing are used – modular, positional and functional. For accounting of minor changes every value is coded with a vector formed by the rules specified below, instead of the only amount.

4.2.2 Modular preprocessing

There is given a set of positive numbers $y_i, ..., y_k$. Calculate each component of the vector Z as follows

$$z_i = \frac{((x \bmod y_i) + y_i)(b - a)}{2y_i} + a ,$$

where [a,b], as before, – the range of acceptable input signals.

Let us explain that when x mod y is calculated the remainder on dividing x by y is returned, for example, if x=5 y=3, x mod y= 2

4.2.3 Functional preprocessing

In a general case, transformation of an input characteristic x into a k-dimensional vector Z occurs as described below.

There are selected k numbers, satisfying the condition

$$x_{\min} < y_i < ... < y_k < x_{\max},$$

The elements of the vector Z are computed

$$z_i = \frac{(\varphi(x - y_i) - \varphi_{\min})(b - a)}{\varphi_{\max} - \varphi_{\min}} + a,$$

where φ - the function, determined within the interval $[x_{\max} - y_k, x_{\min} - y_1]$, a $\varphi_{\max}, \varphi_{\min}$ - the maximal and the minimal value of the function within this interval.

4.2.4 Positional preprocessing [59]

The approach in this case is approximately the same as in case of building the positional representation. A positive y value is selected, meeting the condition $y^k \geq (x_{\max} - x_{\min})$. Let us shift an x parameter so that it had only positive values and zero. For calculation of the vector Z we use the following formulas

$$z_o = (x - x_{\min}) \bmod y,$$

$$z_1 = ((x - x_{\min}) / y) \bmod y,$$

$$z_k = ((x - x_{\min}) / y^k) \bmod y.$$

Other methods of input value conversion – raising to the power, rooting, calculation of reciprocal values, exponential curves and logarithms, and also certain combinations of variables - products, quotients, etc. that can reduce the length of the input data vector.

5 Machine learning in Big Data management

The main problem of ML techniques application in classifycation and regression of big data are computational complexity of the cost function calculation (Equation (2.1)) and of appropriate parameters of the function of hypothesis (Equation (2.2)) due to a large number of examples.

For example, the standard gradient descent algorithm is an iterative procedure provided via comparatively large number of elementary steps.

Let us assume that we have 100 000 000 examples, 10 parameters and calculate parameters for 500 iterations of gradient descent at the average. The number of required calculations will make $5*10^{11}$. Assuming that a computer is able to calculate 1 mln. iterations of gradient descent per minute, we get about 140 hours for calculation of only one set of parameters of the function of hypothesis. If it is needed to draft a learning curve the time of calculation increases proportionally to the number of curve point and can reach absolutely unsuitable values. Though the use of matrix operations (Equation (2.3)) eliminates iterations, still it becomes impossible due to great computational complexity of derivation of a reciprocal matrix.

To overcome "the curse of dimensionality" for the gradient descent algorithms two algorithms are suggested in this case: the one of Stochastic gradient descent – SGD [60] and the one of mini-batch gradient descent – MBGD [61].

The SGD algorithm functions as follows:

1. In the beginning examples from the train set are randomly rearranged.

2. Over all m examples from the learning set, for each of n parameters a new value of the parameter is calculated. In SGD pseudo-code this can be written as:

```
for iter:=1 to K
    for i:=1 to m
        for j:=1 to n
            θ_j := θ_j − α(h_θ(x^(i)) − y^(i)) x_j^(i)
        end
    end
end
```

Number of iterations K=1,...,10.

At the same time, the pseudo-code of the batch gradient descent- BGD algorithm looks as follows

```
Do
    for i:=1 to m
        for j:=1 to n
            θ_j := θ_j − α (1/m) Σ_{i=1}^{m} (h_θ(x^(i)) − y^(i)) x_j^(i)
        end
    end
enddo
```

Thus, at each of m steps of the BGD algorithm cumbersome summing is performed, while SGD uses only one example per each iteration. Assessment of computational complexity of SGD - $O(m)$ comparing to $O(m^2)$ for BGD under condition that the number of parameters n is much fewer than m. Still use of SGD results into approximate decision, not into the global minimum of the cost function. Besides, SGD, same as the described below MBGD also suffer the convergence problem. This means that instead of getting better their result can even get worse with increase in the number of processes examples.

The MBGD algorithm per each iteration uses only a part (b) of examples

```
for iter:=1 to K
    for i:=1 to m with step b
        for j:=1 to n
            θ_j := θ_j − α (1/b) Σ_{k=i}^{k+b} (h_θ(x^(i)) − y^(i)) x_j^(i)
        end
    end
end
```

Its computational complexity can be assessed as $O(bm)$, which in terms of b<<m can be traced to $O(m)$.

Except for the mentioned above SGD and MBGD paralleling of calculations can be applied to overcome the problem of large number of calculations [62, 63], which can allow to decide the problem of finding the parameters of the function of hypothesis with help of BGD during the acceptable period of time parallel to large number of independent processes.

The essence of the Map-reduce method is as follows. Calculation of the sum

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}.$$

Is performed on separate machines or processors. Taken that the number of processors is B pieces, the calculation can be performed like this:

$$Sum_{j0} := \sum_{i=1}^{m/B} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$Sum_{j1} := \sum_{i=1}^{m/B} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

...

$$Sum_{jk} := \sum_{i=k*(m/b)+1}^{m/B+k*(m/B)} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

...

$$Sum_{jB-1} := \sum_{i=(B-1)*(m/b)+1}^{m/B+(B-1)*(m/B)} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} (\sum_{k=0}^{B-1} Sum_{jk})$$

For example, if the number of processors is 3, but the number of examples is 300000:

$$Sum_{j0} := \sum_{i=1}^{100000} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$Sum_{j1} := \sum_{i=100001}^{200000} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$Sum_{j3} := \sum_{i=200001}^{300000} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{300000} (\sum_{k=0}^{3} Sum_{jk})$$

Obviously that computational complexity of the method can be rated as $O(\frac{m^2}{B})$. At small B values, growth of calculation speed will be negligent, but if the number of processors is comparable to the number of processors in modern supercomputer clusters (1000-1000000) the growth rate can reach up to several orders.

Application of ML in big data management has serious opportunities. Special languages and platforms are suggested for realization of ML potential in big data management. For example, in [64] the language (SystemML) is suggested and the ways of its application are described for realization of ML techniques in big clusters MapReduce. In [65] the platform MLBase is described, which provides problem statement and use ML together with high-level operators.

6 Conclusion

The domain of Artificial Intelligence is very vast and includes many disciplines, beginning from logical reasoning to the methods of text tonality analysis. Traditionally recognize the so-called strong artificial intelligence and weak artificial intelligence. The first one is oriented at development of high intelligence human-centric decision systems, eventually at creation of intelligent machines. Weak artificial intelligence is oriented at development of applications, realizing this or another intellectual ability of humans or animals. The potential of weak artificial intelligence concept is realized using machine learning.

The review considers machine learning techniques as a part of weak artificial intelligence methods, applicable for analysis data including for big data processing. The taxonomy of machine learning techniques was proposed. This schema unites various kinds of supervised and unsupervised learning algorithms. At this, classification or other kind of heterogeneous data processing has a range of specific features: heterogeneity of data types, frequently occurring large massives of data, complications in data collation.

The classical diagram of adjustment for machine learning decision techniques is described. The ML problem is formally stated and some frequently used algorithms are described (linear regression, polynomial regression, logistical regression, artificial neural networks, algorithms k-NN, SVN, LDAC, DLDA). Assessment indicators of classification accuracy (accuracy, precision, recall) and the summarizing indicator (T1 Score) are considered in detail. The concept of learn ability of ML algorithms and its practical use (methods of learning curve interpretation) is described. Some data preprocessing methods are described in detail, including the methods of abnormal value elimination and normalization. Briefly considered the application of machine learning methods at the processing of big data and techniques of solving some specific problems with help of parallelized computing, mapreduce approach and modifications of the gradient descent algorithm.

Development of machine learning techniques proceeds in parallel with their practical application, which results into increasing number of applications, appearance special decision techniques of applied problems using ensembles of algorithms. Programming environments and languages including declarative ones are suggested now. Some of them are targeted to facilitate applications at big data. One of the significant task of future researches is development preprocessing methods that could be used at many kinds of data automatically or semi-automatically.

Acknowledgment

This work was supposed by grants 0168/GF4 and 2318/GF3 from the Ministry of Education and Science of the Republic of Kazakhstan.

References

[1] Systems of Neuromorphic Adaptive Plastic Scalable Electronics http://www.darpa.mil/Our_Work/DSO/Programs/Systems_of_Neuromorphic_Adaptive_Plastic_Scalable_Electronics_(SYNAPSE).aspx 10 Aug 2014

[2] Блейксли С, Хокинс Д 2007 *Об интеллекте* 128

[3] Weiß G 1999 *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* 648

[4] Russell S, Norvig P 2010 *Artificial Intelligence: A modern approach* 1078

[5] Gorodetsky V I 2012 Self-organization and multi-agent systems *Proceedings of the Russian Academy of Sciences Theory and control systems* **2** 92-120

[6] Gorodetsky V I 2012 Self-organization and multi-agent systems Applications and technology development *Proceedings of the Russian Academy of Sciences Theory and control systems* **3** 55-75

[7] Jones M T 2008 Artificial Intelligence: A Systems Approach *New Delhi: INFINITY SCIENCE PRESS LLC* 500

[8] Zhang G P 2000 Neural Networks for Classification: A Survey *IEEE Transactions on systems man and cybernetics—Part C: Applications and reviews* **30**(4)

[9] Kriesel D 2015 *A Brief Introduction to Neural Networks* http://www.dkriesel.com/en/science/neural_networks)

[10] Van der Baan M, Jutten C Neural networks in geophysical applications *Geophysics.* **65**(4) 1032-47

[11] Baldwin J L, Bateman R M, Wheatley C L 1990 Application of a neural network to the problem of mineral identification from well logs *The Log Analyst* 279-93

[12] Benaouda B, Wadge G, Whitmark R B, Rothwell R G, MacLeod C 1999 Inferring the lithology of borehole rocks by applying neural network classifiers to downhole logs - an example from the Ocean Drilling Program *Geophysical Journal International* **136** 477- 91

[13] Saggaf M M, Nebrija E L 2003 Estimation of missing logs by regularized neural networks *AAPG Bulletin* **87**(8) 1377-89

[14] Тененёв В А, Якимович Б А, Сенилов М А, Паклин Н Б 2002 Интеллектуальные системы интерпретации геофизических исследований скважин *Штучний інтелект* **3** 338

[15] Алёшин С П, Ляхов А Л 2011 Нейросетевая оценка минерально-сырьевой базы региона по данным геофизического мониторинг *Нові технології* **1**(31) 39-43

[16] Карпенко А Н, Булмасов О В 2015 Применение нейроносетевых технологий при интерпретации данных геофизических исследований скважин. http://oil-gas.platinov-s.com/index.php?name=articles&op=view&id=11&pag=3&num=1

[17] Rogers S J, Chen H C, Kopaska-Merkel D C, Fang J H 1995 Predicting permeability from porosity using artificial neural networks *AAPG Bulletin* 786-1797

[18] Костиков Д В 2007 Инструментальные средства интерпретации геофизических исследований скважин на основе преобразованных каротажных диаграмм с помощью многослойной нейронной сети *Диссертация кандидата технических наук* 189

[19] Muhamediyev R, Amirgaliev E, Iskakov S, Kuchin Y, Muhamedyeva E 2014 Integration of Results of Recognition Algorithms at the Uranium Deposits *Journal of ACIII* **18**(3) 347-52

[20] Амиргалиев Е Н, Искаков С Х, Кучин Я В, Мухамедиев Р И 2013 Интеграция алгоритмов распознавания литологических типов *Проблемы информатики* **4**(21) 11-20

[21] Amirgaliev E, Iskakov S, Kuchin Y, Muhamediyev R 2013 Machine-learning techniques in the pattern recognition of rock at uranium deposits *Proceedings of National Academy of Sciences of Kazakhstan* **3** 82-8

[22] Joseph A C, David S 2006 Wishart Applications of Machine Learning in Cancer Prediction and Prognosis *Cancer Informatics* **2** 59–77

[23] Shoeb A H, John V 2010 Guttag Application of machine learning to epileptic seizure detection *Proceedings of the 27th International Conference on Machine Learning* 975-82

[24] Mannini A, Angelo M S 2010 Machine learning methods for classifying human physical activity from on-body accelerometers *Sensors* **2** 1154-75

[25] Ballester P J, Mitchell J B 2010 A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking *Bioinformatics* **9** 1169-75

[26] Farrar C R, Worden K 2012 *Structural health monitoring: a machine learning perspective* 66

[27] Recknagel F 2001 Application Of macine Learning To Ecological Modelling *Ecological Modelling* 303-10

[28] Charles C, Hecker J, Stuntebeck E, Shea T O 2007 Applications of machine learning to cognitive radio networks *Wireless Communications* IEEE **4** 47-52

[29] Ball N M, Brunner R J 2010 Data mining and machine learning in astronomy *International Journal of Modern Physics* **7** 1049-1106

[30] Szepesvari C 2009 Algorithms for Reinforcement Learning Synthesis Lectures on Artificial Intelligence and Machine Learning series by Morgan & Claypool Publishers 98 http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf

[31] Zhu X 2008 Semi-Supervised Learning Literature Survey Computer Sciences http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf

[32] Kohonen T 1982 Self-Organized Formation of Topologically Correct Feature Maps *Biological Cybernetics* **43**(1) 59–69

[33] Jain A K, Murty M N, Flynn P J 1999 Data Clustering: A Review *ACM Computing Surveys* **31**(3)

[34] Barbakh W A,Wu Y, Fyfe C 2009 Review of Clustering Algorithms. Non-Standard Parameter Adaptation for Exploratory *Data Analysis Studies in Computational Intelligence* **249** 7-28

[35] Ayodele T O 2010 Types of Machine Learning Algorithms *New Advances in Machine Learning* 19-48

[36] Hamza A, Hamza I 2012 Taxonomy of Machine Learning Algorithms to classify realtime Interactive applications *International Journal of Computer Networks and Wireless Communications* (IJCNWC) **2**(1) 69-73

[37] Дьяконов А Г 2010 *Анализ данных, обучение по прецедентам, логические игры, системы WEKA, RapidMiner и MatLab* (Практикум на ЭВМ кафедры математических методов прогнозирования) 277

[38] Møller M F A scaled conjugate gradient algorithm for fast supervised learning *Neural Networks* **6**(4) 525-33

[39] Dong C. Liu, J 1989 Nocedal On the limited memory BFGS method for large scale optimization *Mathematical Programming* **45**(1-3) 503-28

[40] McCulloch W S, Pitts W 1943 A logical calculus of the ideas immanent in nervous activity *The bulletin of mathematical biophysics* **5**(4) 115-33

[41] Rosenblatt F 2015 The perceptron: A probabilistic model for information storage and organization in the brain *Psychological Review* **65**(6) 386-408 http://dx.doi.org/10.1037/h0042519/

[42] Minsky M 1987 Seymour Papert *Perceptrons expanded edition* 308

[43] Минский М 1971 Пейперт С *Персептроны* 263

[44] Minsky M 1987 Seymour Papert *Perceptrons expanded edition* 308

[45] Werbos P 1974 Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences

[46] Werbos P J 1988 Backpropagation: past and future *IEEE International Conference on Neural Networks* **1** 343-53

[47] Saad D 2009 1998 Introduction. On-Line Learning in Neural Networks *Cambridge University Press* 3-8

[48] Cybenco G 1989 Approximation by superpositions of a sigmoidal function *Mathematics of Control Signals and Systems* **2**(4) 304-14

[49] Hornik K 1989 Multilayer feedforward networks are universal approximators *Neural Networks* **2** 359-66

[50] Галушкин А И 2006 Решение задач в нейросетевом логическом базисе *Нейрокомпьютеры: разработка, применение* **2** 49-71

[51] Галушкин А И 2010 Нейронные сети: основы теории *Горячая линия*

[52] Ясницкий Л Н 2008 *Введение в искуственный интеллект: Уч.пос.для вузов* 176

[53] *Нейрокомпьютеры: Учеб. Пособие для вузов 2004* 320

[54] Dudani S A 1976 The distance-weighted k-nearest-neighbor rule

Systems, Man and Cybernetics, IEEE Transactions **4** 325-7
[55] K-nearest neighbor algorithm Support vector machine. http://en.wikipedia.org/wiki/Support_vector_machine 22.02.2012http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm/ 5 Jun 2012.
[56] Support vector machine. http://en.wikipedia.org/wiki/Support_vector_machine 22.02.2012 17 Mar 2013
[57] Linear discriminant analysis. http://en.wikipedia.org/wiki/Linear_discriminant_analysis/ Nov2012
[58] Dudoit S, Fridlyand J, Terence P 2002 Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data *Journal of the American Statistical Association* **97**(457) 77-87
[59] Миркес Е М 1998 *Нейрокомпьютер. Проект стандарта* http://pca.narod.ru/MirkesNeurocomputer.htm 19 Jan 2015.
[60] Bottou L 2010 Large-Scale Machine Learning with Stochastic Gradient Descent *Proceedings of COMPSTAT'2010* 177-86
[61] Bottou L 1998 Online learning and stochastic approximation. On-Line Learning in Neural Networks *Cambridge University Press* 9-43
[62] Chu C-T 2006 Map-Reduce for Machine learning on multicore**.** Advances in Neural Information Processing Systems *Proceedings of the 2006 Conference* 281-310
[63] Leskovec J, Rajaraman A, Ullman J D 2014 Mining of Massive Datasets *Cambridge University Press* 476
[64] Amol G, Krishnamurthy R, Pednault E, Reinwald B, Sindhwani V, Tatikonda S, Tian Y, Vaithyanathan S 2011 SystemML Declarative machine learning on MapReduce *Data Engineering (ICDE) IEEE 27th International Conference* 231-42
[65] Tim K, Talwalkar A, Duchi J C, Griffith R, Franklin M J, d Jordan M I 2013 MLbase: A Distributed Machine-learning System *Conference on Innovative Dta Systems Research (CIDR)* 7-9

Author

Ravil I Muhamedyev, Kazakhstan, Almaty

Current position, grades: Head of Department CSSE&T, professor, International IT University, Almaty, Kazakhstan
University studies: International IT University, Almaty, Kazakhstan
Scientific interest: stochastic simulation, machine learning, simulation of anisochronous systems
Publications: 160 papers, 5 books

Review Paper