Нечеткая кластеризация на основе облачных и туманных вычислениий

М. С. Куприянов[1], И. И. Холод[2], А. В. Шоров[3]
Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
[1]mikhail.kupriyanov@gmail.com, [2]iiholod@mail.ru, [3]ashxz@mail.ru

***Аннотация*. В докладе рассматриваются современные подходы к интеллектуальному анализу Больших данных на основе технологий облачных и туманных вычислений. Рассматриваются существующие решения, их достоинства и недостатки. Предлагается новый подход выполнения интеллектуального анализа в рамках концепции туманных вычислений и его применения для Интернета вещей. Описываются особенности его применения к решению задачи нечеткой кластеризации.**

Ключевые слова: нечеткая кластеризация; интеллектуальный анализ; распределенный анализ; облачные вычисления; туманные вычисления; Интернет вещей

I. Введение

В последнее время у человечества остро встала проблема анализа накоплены данных, которые содержат полезные знания. С развитием технологии Интернета вещей (Internet of Things), источниками информации становятся не только люди, но и устройства (сенсоры, смартфоны, видеокамеры и т.п.) подключаемые к сети. Они формируют потоки разнородных данных, увеличивая количество распределенных источников информации. Лавинообразный рост источников и объемов информации, ее разнородность и распределенный характер хранения привели к необходимости пересмотра технологий обработки данных.

Одной из важных задач анализа данных является кластеризация. В условиях неопределенности в анализируемой среде актуальной является задача нечеткой кластеризации. Алгоритмы нечёткой кластеризации представляют большой интерес для тех случаев анализа, когда данные описывают объекты или процессы в условиях неопределенности и реализуют обычно итерационную процедуру приближения к решению задачи.

В настоящее время для обработки Больших данных широко используются облачные вычисления в интеграции с технологией распределенной обработки данных – Map Reduce. Такая обработка предполагает сбор всей информации в хранилище данных, возможно виртуальном, однако являющимся единым источником для аналитических сервисов. Это приводит к следующим проблемам:

- затрачивается время на передачу информации, что может быть критично при ее обработке, например, в режиме реального времени;
- увеличивается сетевой трафик, что снижает возможности использования каналов связи с низкой пропускной способностью (спутниковых каналов связи, беспроводных каналов и т.п.);
- требуется передача информации, в том числе конфиденциальной, по открытым каналам передачи данных;
- повышается ценность собранной в одном месте информации, что требует повышенных мер обеспечения ее безопасности и надежности.

Исключить указанные недостатки можно за счет анализа непосредственно на источниках информации. Подобный подход соответствует концепции «туманных» вычислений (Fog Computing) [1]. Однако для реализации данной концепции в области анализа данных необходимо построение параллельных алгоритмов, позволяющих выполнять анализ в модели с распределенной памятью. Кроме того, новые методы и средства должны учитывать тип распределения данных: горизонтальный, когда на разных источниках хранится информация о разных объектах и явлениях, и вертикальный, когда на разных источниках хранится информация о разных характеристиках одних и тех же объектов.

В данном докладе рассматривается современные решения в области интеллектуального анализа Больших данных, а также предлагается подход к его выполнению в рамках туманных вычислений и применения в системах Интернета вещей. Рассматриваются особенности его применения к задаче нечеткой кластеризации.

II. Современные средства анализа Больших данных

Рост объемов хранимой информации привел к появлению понятия Больших данных, которые часто определяются тремя V: Volume, Variety и Velocity [2]:

- большие объемы информации (data volume);
- различные форматы (data variety);
- генерируются с высокой частотой (data velocity).

Анализ таких данных является ресурсоемкой задачей. В связи с этим для обработки и анализа Больших данных используют средства распределенных вычислений, реализующие разные модели [3].

Наиболее популярной в настоящее время является программная модель распределенных вычислений MapReduce [4]. Она разделяет обработку данных на две функции, реализуемые разработчиком: map, выполняющую обработку отдельных экземпляров данных, и reduce, выполняющую слияние результатов обработки в общий.

Модель программирования MapReduce успешно применяется для распределенного выполнения алгоритмов интеллектуального анализа данных. Можно выделить ряд современных библиотек таких алгоритмов:

Apache Mahout [5] – библиотека алгоритмов, адаптированных к модели MapReduce, которые могут выполняться на платформе Apache Hadoop [6]. Она содержит ряд алгоритмов для распределенного выполнения: классификации (Naive Bayes, Random Forest, Multilayer perceptron classifier (MLPC)), регрессии (Ordinary Least Squares (OLS), Linear Regression, Support Vector Machine (SVM)), кластеризации (Canopy-clustering). Она также позволяет добавлять собственные алгоритмы, декомпозированные на функции map и reduce.

Apache Spark Machine Learning Library (MLlib) [7] библиотека алгоритмов machine learning, адаптированных к модели MapReduce, которые могут выполняться на платформе Apache Spark. Она содержит следующие алгоритмы: классификации (Logistic Regression, Random Forest, MLPC, Naive Bayes), регрессии (Linear Regression, SVM), кластеризации (K-Means), поиска частых наборов (FP-Growth). Пользователь также может расширять данную библиотеку.

ML Grid [8] из **Apache Ignite 2.7.0** содержит алгоритмы machine learning для распределенного выполнения. Она включает в себя следующие алгоритмы: Linear Regression, K-Means, MLPC, Fuzzy C-Means и k-NN (k-nearest neighbors). Они могут выполняться на платформе Ignite Compute Grid, которая также реализует модель MapReduce. Библиотека также является расширяемой.

Для анализа потоковых Больших данных может быть использована платформа **Scalable Advanced Massive Online Analysis (SAMOA)** [9]. Она включает в себя распределенные алгоритмы: Vertical Hoeffding Tree (VHT), Very Fast Decision Tree (VFDT), CluStream, Adaptive Model Rules (AMRules), PARMA. SAMOA имеет адаптеры для выполнения алгоритмов на различных платформах, реализующих модель MapReduce. SAMOA позволяет разработчикам интегрировать существующие алгоритмы из библиотеки MOA.

ТАБЛИЦА I СРАВНЕНИЕ БИБЛИОТЕК РАСПРЕДЕЛЕННЫХ АЛГОРИТМОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Свойства	**Apache Mahout**	**Apache Spark MLib**	**ML Grid Apache Ignite**	**SAMOA**	**Vowpal Wabbit**
Модель	MapReduce	MapReduce	MapReduce	MapReduce	MapReduce
Платформа	Apache Hadoop	Apache Spark (Akka – Actor model)	Apache Ignite	S4 (Actor model)	Apache Hadoop
Источник данных	Один	Один	Один	Один	Один
Задачи анализа	Классификация, Регрессия, Кластеризация	Классификация, Регрессия, Кластеризация,	Classification, Regression, Clustering	Classification, Clustering, Regression streams	Classification, Regression
Кол-во алгоритмов	7	8	5	5	9
Добавление новых	Да	Да	Да	Да	Да
Номер версии	0.13.0	2.3.0	2.4.0	0.4.0	8.5.0
Дата релиза	Дек 2017	Фев 2018	Март 2018	Авг 2016	Дек 2017

ТАБЛИЦА II СРАВНЕНИЕ ОБЛАЧНЫХ ПЛАТФОРМ

Характеристики	**Azure ML**	**Amazon ML**	**GoogleCloud ML**	**Watson Analytics**
Облачная модель	SaaS	SaaS	SaaS	SaaS
Интерфейс пользователя	Web	Web	API	Web
Пользовательский уровень	Аналитик	Аналитик	Разработчик	Аналитик
API	REST	REST	REST	REST
Масштабируемость	Да	Да	Да	Да
Место хранения данных	Внутри облака	Внутри облака	Внутри облака	Внутри облака
Платформа распределенных вычислений	Vowpal Wabbit (Hadoop)	Apache Hadoop	Apache Hadoop	-
Полный цикл анализа	Да	Да	Да	Нет
Задачи анализа	Классификация, Регрессия, Кластеризация	Классификация, Регрессия	Классификация, Регрессия	Классификация, Регрессия
Добавление новых алгоритмов	Из MLMarketplace	Нет	Нет	Нет
Использование	Платный	Платный	Платный	Платный

Vowpal Wabbit (VW) [10] библиотека алгоритмов анализа, начатая как открытый проект, компанией Yahoo!. В настоящее время она развивается компанией Microsoft. Она использует платформу Apache Hadoop для масштабируемых вычислений и содержит несколько алгоритмов data mining для классификации и регрессии: OLS, Matrix factorization (sparse matrix SVD), Single layer neural net (with user specified hidden layer node count), Searn

(Search and Learn), Latent Dirichlet Allocation (LDA), Stagewise polynomial approximation, One-against-all (OAA), Weighted all pairs, Contextual-bandit.

Таким образом, все современные библиотеки алгоритмов анализа Больших данных используют различные платформы распределенных вычислений, основанных на модели MapReduce. Однако, адаптация алгоритма к данной модели достаточна трудоемка. Следствием сложности адаптации алгоритмов является небольшое количество алгоритмов анализа данных, реализованных в описанных библиотеках (табл. 1).

Учитывая бурный рост технологий распределенных вычислений и облачных вычислительных сред, естественным является интеграция технологий анализа данных, распределенных и облачных вычислений. В настоящее время существуют системы, которые можно было бы отнести к облачным технологиям анализа данных.

Azure Machine Learning (Azure ML) [11] – SaaS облачный сервис от компании Microsoft Inc. Он был запущен в Феврале 2015 года. Azure ML и обеспечивает платные сервисы, которые позволяют пользователям выполнять полный цикл анализа: сбор данных, подготовку к анализу, настройку, анализ данных и оценку результатов. Сервис ориентирован на пользователей со знаниями в области анализа данных.

Azure ML может выполнять анализ данных, которые заранее размещены в облаке. Для этого оно содержит различные средства для импорта данных. Для анализа пользователь может использовать только заранее заданные алгоритмы: classification (Boosted Decision Trees, Random Forests, Logistic Regression, SVM, Averaged Perceptron, neural networks), regression (Linear Regression, Boosted Decision Trees, neural networks), anomaly detection (SVM, PCA) and clustering (K-Means). Дополнительные алгоритмы доступны на Machine Learning Marketplace.

В апреле 2015 года компания Amazon выпустила на рынок свое решение для облачного анализа данных **Amazon Machine Learning** (Amazon ML). Оно представляет собой сервис для обучения предсказательных моделей [12]. Сервис обеспечивает все необходимые стадии анализа: подготовку данных, построение модели, настройку, оценку модели и др. Для использования Amazon ML пользователю не требуются специальные знаний в области анализа.

Amazon ML решает только задачи классификации и регрессии. Он включает в себя алгоритмы: binary classification, multiclass classification, and regression. Новые алгоритмы не могут быть добавлены пользователем самостоятельно. Как и в случае с Azure ML, сервис Amazon ML позволяет анализировать данные, размещенные только внутри облака. Для масштабирования вычислений используется Apache Hadoop.

Компания Google в марте 2016 года представила свою платформу **Cloud Machine Learning** (Cloud ML) [13], которая используется для анализа фотографий, переводов и почты. Она использует алгоритмы машинного обучения – нейронные сети. Данный сервис Google обеспечивает REST API для распознавания образов, речи, языковых переводов и т.п. Сервис также не позволяет расширять состав алгоритмов анализа.

В начале 2016 года компания IBM выпустила интеллектуальный аналитический сервис **Watson Analytics** [14]. Он решает высокоуровневые аналитические задачи, взаимодействуя с пользователями посредством запросов на естественном языке. Watson Analytics анализирует данные, размещенные в облаке. Пользователи могут использовать только те алгоритмы и методы, которые уже заложены в сервис.

Основными недостатками описанных систем являются (табл. 2):

1) необходимость хранения анализируемых данных внутри облака, что в свою очередь имеет ряд недостатков:

- требует дополнительных аппаратных средств для хранения данных;
- для обработки актуальных данных нужно или всегда хранить их во внутреннем хранилище или решать задачу их синхронизации с источником информации;
- при загрузке информации производится преобразование во внутренние форматы, что может привести к искажению информации и/или вызывать ошибку при загрузке;
- обеспечение конфиденциальности хранящейся во внутреннем хранилище информации ложится на провайдера сервиса, что не всегда может удовлетворить владельца информации;
- не использует преимуществ работы с общей памятью.

2) привязка только к одной технологии выполнения распределенных вычислений (в основном Map Reduce и ее реализация на Apache Hadoop), каждая из которых имеет свои недостатки и может эффективно применяться только при определенных условиях.

Также, необходимо заметить, что ни одна из систем не решает задачу нечеткой кластеризации.

Использование облачных сервисов для выполнения анализа в системах с распределенными источниками, например, в системах Интернета вещей, приводит к необходимости передачи данных по сети, увеличивая трафик и задержку в обработке. Чтобы избежать этих недостатков была предложена технологии туманных вычислений (Fog computing) [1]. Она предполагает наличие вычислительных узлов (Fog узлы) на уровне устройств и промежуточном уровне IoT-систем, на которых выполняется анализ данных без их пересылки в централизованное хранилище.

Популярность подобных архитектур обусловлена отсутствием в них недостатков обозначенных выше. Туманные вычисления полностью решают или снижают влияние ряда проблем в распределенных системах: высокая задержка в сети; масштабирование источников

информации; трудности, связанные с подвижностью конечных точек; высокая стоимость полосы пропускания; территориальная распределенность систем.

Несмотря на достоинства и популярность идеи туманных вычислений, готовые решения для ее реализации в настоящее время отсутствуют. Это объясняется как недостаточной проработкой данной концепции, так и высоким уровнем абстракции ее представления. Кроме того, необходимо учитывать и способ распределения данных между источниками: вертикальный или горизонтальный.

III. Распараллеливание алгоритмов анализа для выполнения в концепции туманных вычислений

Для распараллеливания алгоритмов интеллектуального анализа данных и их выполнения в рамках концепции туманных вычислений, мы использовали принципы функциональной парадигмы вычислений. Для этого мы представили алгоритм анализа данных в виде функции, принимающей в качестве входного аргумента набор данных $d \in D$ и возвращающей в качестве результата модель знаний $m \in M$:

$$f: D \rightarrow M. \quad (1)$$

Набор данных, обычно представляют матрицей вида:

$$d = (x_{j.k})_{j=1,k=1}^{z,p} = \begin{pmatrix} x_{1.1} & \dots & x_{1.k} & \dots & x_{1.p} \\ \dots & & \dots & & \dots \\ x_{j.1} & \dots & x_{j.k} & \dots & x_{j.p} \\ \dots & & \dots & & \dots \\ x_{z.1} & \dots & x_{z.k} & \dots & x_{z.p} \end{pmatrix}.$$

где $x_{j.k}$ значение k^{ro} атрибута j^{ro} объекта наблюдения. Все множество объектов наблюдения обозначается как X.

В случае распределенного хранения набора данных, он представляет собой части матрицы данных:

$$d = d_1 \cup d_2 \cup \dots \cup d_h \cup \dots \cup d_w, \text{ где}$$

d_h - подматрица-данных, размещаемая на h-м узле источнике:

$$d_1 = (x_{j.k})_{j=1,k=1}^{y,p},\ d_2 = (x_{j.k})_{j=y+1,k=1}^{x,p},\ \dots,\ d_w = (x_{j.k})_{j=x+1,k=1}^{z,p}.$$

Функция (1) может быть декомпозирована на ряд функций:

$$f = f_n \circ f_{n-1} \circ \dots \circ f_s \circ \dots \circ f_r \circ \dots \circ f_1 \circ f_0. \quad (2)$$

В представленной композиции первая функция f_0 принимает в качестве аргумента набор данных $d \in D$ и возвращает созданную по нему модель знаний $m_0 \in M$:

$$f_0: D \rightarrow M.$$

Остальные функции композиции (2) f_r, r=1..n принимают в качестве входного аргумента модель знаний m_{r-1}, построенную предыдущей функцией f_{r-1} и возвращает измененную модель знаний m_r:

$$f_r: M \rightarrow M.$$

Функции, которым для изменения модели знаний требуются данные, дополнительно в качестве первого аргумента принимают набор данных $d \in D$. Они имеют тип:

$$f_r: D \rightarrow M \rightarrow M.$$

Представление алгоритма анализа данных в виде функционального выражения (2) упрощает его распараллеливание. Для этого мы ввели функцию высшего порядка:

parallel : Boolean → [(M → M)] → M → M.
parallel s [f_1, …, f_r] m = if (s == true) head fork [f_1, …, f_r] m
else join m (fork [f_1, …, f_r] m), где

head – функция, возвращающая первый элемент списка.

Функция высшего порядка fork, принимает список функций и модель знаний, применяет каждую функцию из списка к модели знаний и полученную модель знаний добавляет в возвращаемый список:

fork : Boolean → [M → M] → M → [M].
fork s [f_1, …, f_r] m = if (s == true) [f_1 m, …, f_r m] m
else [f_1 clone m, …, f_r clone m].

В функциях parallel и fork первый булевский аргумент s определяет способ выполнения: с общей или распределенной памятью. В результате распараллеленный алгоритм интеллектуального анализа данных будет выполняться в соответствии с концепцией туманных вычислений, так как это показано на рис. 1.

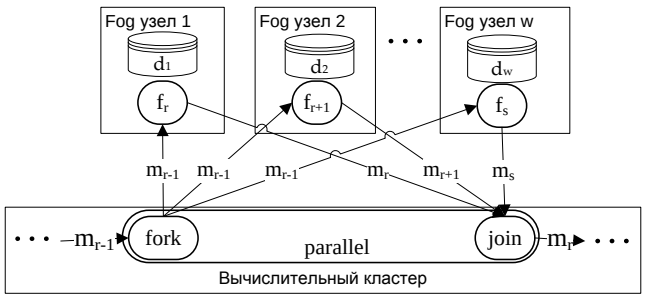

Рис. 1. Выполнение алгоритма анализа в соответствии с концепцией туманных вычислений

IV. Распараллеливание алгоритмов нечеткой кластеризации для туманных вычислений

A. *Алгоритмы нечёткой кластеризации*

В задаче нечеткой кластеризации направление решения и окончание процесса его поиска задаёт целевая функция:

$$J(X,U,C) = \sum_{i=1}^{|C|}\sum_{j=1}^{|X|} u_{ij}^{w}\delta^2\left(x_j, c_i\right),$$

где U – матрица принадлежности, C – множество центров кластеров; x, c, u – элементы соответствующих множеств; w – показатель нечёткости, δ – метрика расстояния. Алгоритм осуществляет минимизацию целевой функции следующим образом:

1. определение мощности множества кластеров;
2. определение метрики расстояния;
3. определить критерий остановки;
4. инициализировать матрицу разбиения (например, случайны образом);
5. вычислить центры кластеров по формуле:

$$c_i = \frac{\sum_{j=1}^{|X|} \left(u_{ij}\right)^w x_j}{\sum_{j=1}^{|X|} \left(u_{ij}\right)^w}, \quad 1 \le i \le |C|; \qquad (3)$$

6. обновить матрицу принадлежности по формуле:

$$u_{ij} = \frac{1}{\sum_{k=1}^{|C|} \left(\frac{\delta^2(x_j, c_k)}{\delta^2(x_j, c_k)} \right)^{\frac{1}{w-1}}}, \qquad (4)$$

для всех $1 \le i \le |C|, \quad 1 \le j \le |X|$;

7. проверить критерий остановки, если выполняется – завершить, иначе перейти к шагу 5.

Это типичная процедура, используемая в алгоритмах нечёткой кластеризации, тем не менее, важно отметить, что минимизация целевой функции – не единственный способ построения разбиения на кластеры.

B. Распараллеливание алгоритма нечеткой кластеризации для туманных вычислений

В соответствие с предложенным подходом выполнения алгоритмов интеллектуального анализа в концепции туманных вычислений, алгоритм нечеткой кластеризации должен быть представлен в виде композиции функций (2). Для этого каждый шаг алгоритма может быть представлен функцией:

$$f = f_7 \circ f_6 \circ f_5 \circ f_4 \circ f_3 \circ f_2 \circ f_1 \circ f_0$$

Для их выполнения в рамках концепции туманных вычислений необходимо выделить функции непосредственно взаимодействующие в данными X. Такими функциями являются f_5 (3) и f_6 (4). Для их переноса на источники данных, необходимо их разделить на функцию обрабатывающие данные и функцию, использующую результат такой обработки.

Для функции (3) такое разделение выполняется достаточно легко:

$$f_5 = f_5'' \circ f_5', \text{ где } f_5' = \sum_{j=1}^{|X|} \left(u_{ij}\right)^w x_j, \; f_5'' = \frac{f_5'}{\sum_{j=1}^{|X|} \left(u_{ij}\right)^w}.$$

При этом функция f_5' легко распределяется между источниками данных функцией parallel. Результаты распределенной обработки суммируются, функцией join.

Для функции (4) такое разделение выглядит следующим образом:

$$f_6 = f_6'' \circ f_6', \text{ где } f_6' = \sum_{k=1}^{|C|} \left(\frac{\delta^2(x_j, c_k)}{\delta^2(x_j, c_k)} \right)^{\frac{1}{w-1}}, \; f_6'' = \frac{1}{f_5'}.$$

Функция f_6' также распределяется между источниками данных функцией parallel, а ее результаты суммируются функцией join.

В результате алгоритм нечеткой кластеризации для выполнения согласно концепции туманных вычислений будет записан следующим образом:

$$f = f_7 \circ f_6'' \circ \text{parallel}(f_6') \circ f_5'' \circ \text{parallel}(f_5') \circ f_4 \circ f_3 \circ f_2 \circ f_1 \circ f_0$$

Распределение функций между узлами изображено на рис. 2.

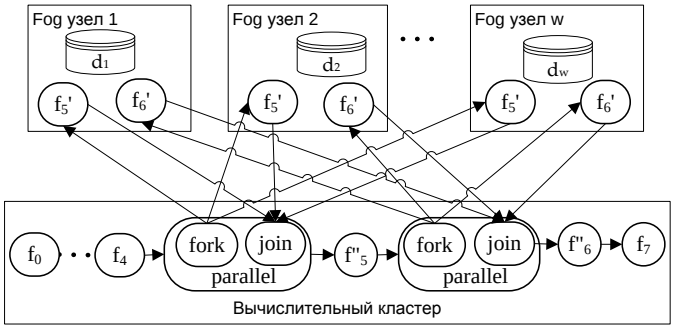

Рис. 2. Выполнение алгоритма нечеткой кластеризации в соответствии с концепцией туманных вычислений

V. ЗАКЛЮЧЕНИЕ

В работе рассмотрены современные подходы к обработке Больших данных, в том числе генерируемых в системах Интернета вещей. В настоящее время для этого широко используются облачные технологии. Ведущие компании предлагают собственные облачные сервисы, для анализа данных пользователей, в том числе данных поступающих от систем Интернета вещей. Такие решения используют библиотеки алгоритмов анализа, реализующую модель распределенных вычислений MapReduce. Среди прочих недостатков такого подхода, он предполагает сбор данных в облаке в едином хранилище. Это в свою очередь ведет к увеличению сетевого трафика, задержкам в обработке информации и снижению безопасности данных.

Для устранения указанных недостатков был предложен подход построения параллельных алгоритмов анализа данных, соответствующий концепции туманных вычислений. Данная концепция предполагает перенос части вычислений «ближе» к данным. Для этого алгоритм был представлен в функциональном виде, и была введена специальная функция, позволяющая выполнять параллельный анализ декомпозированного алгоритма, как с общей, так и с распределенной памятью.

Рассмотрено применение предложенного подхода к распараллеливанию алгоритма нечеткой кластеризации в соответствии с концепцией туманных вычислений. Определены функции, выполняющиеся на узлах источниках и их композиция для такого выполнения.

СПИСОК ЛИТЕРАТУРЫ

[1] Bonomi F. Fog Computing and its Role in the Internet of Things [Text] / R. Milito, J. Zhu, S. Addepalli // Proceeding of MCC. 2012, pp. 13-16.

[2] Laney D. 3-D Data Management: Controlling Data Volume, Velocity and Variety. META Group Research Note, Stamford. [Text]/ D. Laney - META Group Inc, 2001 – (https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf)

[3] Топорков В.В. Модели распределенных вычислений [Текст] / В.В. Топорков. М.: ФИЗМАТЛИТ, 2004. 320 с.

[4] Dean J. MapReduce: Simplified data processing on large clusters [Text]/ J. Dean, S. Ghemawat // Proceedings of Operating Systems Design and Implementation. NY.:ACM, 2004. p. 107–113.

[5] Ingersoll, G. Introducing Apache Mahout [Text]/ G. Ingersoll – (http://www.ibm.com/developerworks/java/library/j-mahout/)

[6] Apache Ignite. Documentation. Machine Learning. – (https://apacheignite.readme.io/docs/machine-learning)

[7] Machine Learning Library (MLlib) Guide. – (http://spark.apache.org/docs/latest/mllib-guide.html)

[8] Apache Ignite. Documentation. Machine Learning. – (https://apacheignite.readme.io/docs/machine-learning)

[9] De Francisci Morales, G. SAMOA: Scalable Advanced Massive Online Analysis [Text]/ G. De Francisci Morales, A. Bifet // Journal of Machine Learning Research. 2015. vol. 16. p. 149−153.

[10] Langford J. Vowpal wabbit [Text]/ J. Langford, A. Strehl, L. Li. 2007. (http://hunch.net/ ~vw).

[11] Gronlund C. J. Introduction to machine learning on Microsoft Azure. A guide for technical professional [Text]/ C.J. Gronlund. Chappell & Associates, 2015 – (https://azure.microsoft.com/en-gb/documentation/articles/machine-learning-what-is-machine-learning/).

[12] Barr J. Amazon Machine Learning – Make Data-Driven Decisions at Scale. Amazon Machine Learning [Text]/ J. Barr. 2016. (https://aws.amazon.com/ru/blogs/aws/amazon-machine-learning-make-data-driven-decisions-at-scale/).

[13] Google Cloud Machine Learning at Scale. (https://cloud.google.com/products/machine-learning/).

[14] Lally A. Question analysis: How Watson reads a clue [Text]/ Lally A. et. al. // IBM Journal of Research and Development. 2012. vol. 56, n.3.4. p. 1-2.