HAL
open science

Resource-aware distributed knowledge discovery

João Gama, Antoine Cornuéjols

To cite this version:

João Gama, Antoine Cornuéjols. Resource-aware distributed knowledge discovery. Michael May; Lorenza Saitta. Ubiquitous Knowledge Discovery, 6202, Springer, 2010, Lecture Notes in Computer Science, 978-3-642-16391-3. 10.1007/978-3-642-16392-0_3 . hal-02480266

HAL Id: hal-02480266

https://hal.science/hal-02480266v1

Submitted on 15 Feb 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

2 WG3 - Resource Aware Distributed Knowledge Discovery

João Gama and Antoine Cornuéjols

2.1 The challenge of ubiquitous computing

Two major technological evolutions have the potential to deeply modify our relationship with our environment:

- With the advent of widely available and cheap computer power, inanimate things are coming to life. **Simple objects** that surround us are gaining sensors, computational power, and actuators, and **are changing from static, inanimate objects into adaptive, reactive systems** that have thus the potential to become more useful and efficient.

Smart Devices

- In parallel, the **explosion of networks of all kinds**, from long distance high bandwidth ones to local low bandwidth wifi systems, offers new unknown possibilities for the development and self-organization of communities of intelligent communicating appliances.

Networks

It is now possible to envision having intelligent appliances and, in general, ambient intelligence aimed at easing our life. [29] gives illustrative and futurist scenarios. But this requires to know how to embed these new heterogeneous appliances with the right kind of sensors and of intelligence in order to give them the capability to sense their environment and be able to communicate and cooperate, while at the same time be able to interact gracefully and naturally with human users. Furthermore, these intelligent items, and the networks they will be part of, require the ability to **adapt continuously to ever changing environmental conditions and needs** of the users. The pervasiveness of mobile phones in the population attracts the attention of the actors in this sector to the wealth of data that could be extracted from simple measurements automatically sent by the phones during their lifetime.

It is therefore clear that a wide spectrum of learning abilities is in demand for these new developments to take place. First, smart devices adaptable to a diversity of users and ready to accompany them for years, will need to:

- be able to *sense their environment and receive data from other devices*, and make sense of the gathered data. This is related to feature selection, data cleaning and data fusion issues.

Sensing and Acting

- be able to *adapt continuously to changing environmental conditions* (including their own condition) and evolving user habits and needs. This touches both short-term and real-time adaptiveness and longer term capability for incremental learning and changes detection.

Adaptation

- be capable of predictive *self-diagnosis*. A significant and useful intelligence characteristic is diagnostics - not only after failure has occurred,

but also predictive (before failure) and advisory (providing maintenance instructions). The development of such self-configuring, self-optimizing, and self-repairing systems is a major scientific and engineering challenge. To meet this challenge, Autonomic Computing requires extensive use of AI techniques such as automated real-time reasoning and decision making, machine learning, and planning. Thus, Autonomic Computing promises to be a major application area for AI, a driver for basic research, and a cross-pollinator across many sub-fields of AI.

Sel-Diagnosis

- *be resource-aware* because of the *real-time constraint and of limited computer, battery power and communication resources*. Indeed, the development of ambient intelligence forces us to enter the world of limited rationality where the availability of time, computer and communication resources must enter the reasoning process.

Resource awareness

The last twenty years or so have witnessed large progress in machine learning and in its capability to handle real-world applications. Nevertheless, **machine learning so far has mostly centered on one-shot data analysis from homogeneous and stationary data, and on centralized algorithms**. Nowadays we are faced with **tremendous amount of distributed data** that could be generated from the ever increasing number of smart devices. In most cases, **this data is transient**, and may not be stored in permanent relations. A large part, if not the whole of **the theory of machine learning (e.g. PAC learning model), relies on the assumption that the data points are independent and identically distributed, meaning that the underlying generative process is stationary**.

Example: Forecast of Electrical Load

An illustrative Problem. Electricity distribution companies usually set their management operators on SCADA/DMS products (Supervisory Control and Data Acquisition / Distribution Management Systems). One of their important tasks is to forecast the electrical load (electricity demand) for a given subnetwork of consumers. Load forecast is a relevant auxiliary tool for operational management of an electricity distribution network, since it enables the identification of critical points in load evolution, allowing necessary corrections within available time, and planning strategies for different horizon.

In this context, data is collected from a set of sensors distributed all around the network. Sensors can send information at different time scales, speed, and granularity. Data continuously flow eventually at high-speed, in a dynamic and time-changing environment. Data mining in this context requires continuously processing of the incoming data monitoring trends, and detecting changes. Traditional one-shot systems, memory based, trained from fixed training sets and generating static models are not prepared to process the high detailed data available, they are not able to continuously maintain a predictive model consistent with the actual state of the nature, nor are they ready to quickly react to changes. Moreover, with the evolution of hardware components, these sensors are acquiring computational power. The challenge will be to run the predictive model in the sensors itself.

2.1.1 A World in Movement.

The constraints we have enumerated implies to switch **from one-shot single-agent learning tasks to a lifelong and spatially pervasive perspective**. In the novel lifelong perspective induced by ubiquitous environments, **finite training sets, static models, and stationary distributions will have to be completely thought anew**. In that context, Data Mining approaches involving fixed training sets, static models and evaluation strategies are obsolete.

All these aspects entails the new characteristics for the data:

Data Characteristics

- Data are made available through *unlimited streams* that continuously flow, eventually at high-speed, over time;
- The underlying *regularities may evolve over time* rather than be stationary
- The data can no longer be considered as *independent and identically distributed*.
- The data is now often *spatially as well as time situated*.

But does the existence of ubiquitous environments really change the problem of machine learning? Wouldn't simple adaptations to existing learning algorithms suffice to cope with the new needs described in the foregoing? These new concerns might indeed appear rather abstract, and with no visible direct impact on machine learning techniques. Quite to the contrary, however, **even very basic operations that are at the core of learning methods are challenged in the new setting**. For instance, consider the standard approach to cluster variables (columns in a work-matrix). In a batch scenario, where all data is available and stored in a working matrix, we can apply any clustering algorithm over the *transpose* of the working matrix. In a scenario where data evolves over time, this is not possible, because the transpose operator is a blocking operator [4]: the first output tuple is available only after preocessing all the input tuples. Now, think of the computation of the entropy of a collection of data when this collection comes as a data stream which is no longer finite, the domain of variables can be huge, and where the number of classes of objects is not known a priori; or think on continuous maintenance the k-most frequent items. And then, what becomes of statistical computations when the learner can only afford a one pass on each data piece because of time and memory constraints and has to decide on the fly what is relevant and must be further processed and what is redundant or not representative and could be discarded. These are but two examples of a clear need for new algorithmic approaches in the KDubiq framework.

Data Mining is thus faced with new challenges. All of them share common issues: distributed continuously flow of data generated by evolving distributions, the domains involved (the set of attribute-values) can be also huge, and computation resources (processing power, storage, bandwidth, and battery power) are limited.

In short, machine learning algorithms will have to enter the world of **limited rationality**, e.g. rational decisions are not feasible in practice due to the finite computational resources. To reshape, ubiquitous data mining implies new requirements to be considered. The new constraints include:

Limited Resources

- The algorithms will have to use *limited computational resources* (in terms of computations, space and time)
- The algorithms will have only a *limited direct access to data* and may have to communicate with other agents on *limited bandwidth* resources.
- In a community of smart devices geared to ease the life of users in real time, answers will have to be ready in an *anytime protocol*.
- Overall, data gathering and data (pre-)processing will be *distributed*.

In this chapter we discuss algorithm issues related to advanced data analysis in dynamic environments using devices with limited resources. In the contexts we are interested in data is distributed data, flowing eventually at high-speed. We identify the limitations of the current Machine Learning theory and practice and identify the most desirable characteristics for a learning algorithm in these scenarios. The analysis follow three main dimensions: learning in resource aware devices, distributed data, and continuous flow of data. These dimensions define the organization of the chapter.

2.2 The Challenge of Limited Resources

Ubiquitous environments mark the end of the area of the single and unlimited learning agent paradigm. The fact that a multitude of simple and situated agents are now to form a new kind of anytime responsive system implies a profound reexamination of the data processing and data mining tasks. The main point here is the **degree of autonomy** of the agents. We would like to maximize the level of autonomy, requiring the ability to process their own data, and make local decisions. Most important, agents might communicate with neighbors, minimizing communication costs, to make collective and rational decisions.

Autonomy

Traditional data mining applications call for an off-line centralized data analysis processing over an existing (very) large database. In Ubiquitous Data Mining (UDM), every one of the assumptions that underlie this picture are challenged. Indeed, in this setting, *data is transient*, obtained through a web of, possibly heterogeneous, sensors with limited computational, memory and communication bandwidth capabilities. Furthermore, both for reasons of scarce resources and because of real-time demands, data must be processed, at least partly, on the fly and locally. Applications in fusion of environmental sensor measures, in the analysis of information gathered by web/blog crawlers that independently sip through collections of texts and links, in analysis and control of distributed communication networks have been already cited. But many more are coming fast in scientific, business and industrial contexts.

These new constraints challenge the existing data mining techniques and require a significant research effort to overcome their current limitations.

Resource Limitation

First of all, *at the level of each individual device or agent*, the limitations on both computing power, memory space and time will constrain the type of processing that can be done. For instance, floating point operations may be unavailable, and processing power may be severely limited beyond the one required to run the normal operations of the device. Therefore, algorithms for local data mining must require very low complexity processes, possibly at the cost of degraded output. In many settings, sensor nodes and computing units (e.g. handheld devices) lack sufficient memory to run classical data mining techniques which require that the results of the intermediate processing steps be resident in memory over the processing time of the running algorithm. Data, often coming in the form of continuous streams, will have either to be partitioned into subsets small enough as to be tractable, or processed on a one-pass only basis. Furthermore, the locally available data may be too restricted to permit an informed enough data mining process, and queries will have to be addressed to other agents. Similarly, it is possible to envision cases where one device will have to recruit the computing power of other less busy agents to meet its own needs. In all these cases, this will require that the individual devices have sufficient self-monitoring capabilities so as to know when to ask for additional information or for help in the form of computing power.

Integration

Secondly, *at a more integrated level*, several issues will need to be addressed. To begin with, at quite a basic level, the heterogeneity of the individual devices will introduce problems of communication when the set of measurements and the formats in which they are expressed will differ among agents. For instance, in sensor networks, work on a sensor model language called sensorML has been started as part of the Opengeospatial Consortium. It uses XML to encode and describe sensor models. The whole process ranging from input to output and associated parameters are described using this language. Apart from the issue of heterogeneousness of the devices, the whole system will have to be responsive even in face of faulty or missing measurements, drifting characteristics of the sensors and generally haphazardous malfunctions. Furthermore, measured data might well be redundant on one side, but also different in their own right because issued from different locations and aimed at different tasks. These characteristics will require that methods for the management of the system be developed. With regards to learning, meta-learning methods will be needed in order to distribute learning processes when needed and to integrate the results on a anytime basis. Since the architecture of the system may well not incorporate a global master device, these management capabilities will have themselves to be distributed and part of the self-aware computing capacities of the individual agents. In this respect, much remains to be done, and we do not have, at this time, a mature methodology to meet that challenge.

Bandwidth Limitation

The limited communication bandwidth between the agents poses other questions. In-network knowledge integration is an open research issue in Ubiquitous Data Mining. It has impact on many problems like: merging of clustering models, of classification decisions and frequent patterns mining. Controlling the

accuracy of the resulted integrated knowledge is fast becoming a critical research question. Exchange of data should not be done without some form of preprocessing that clean the data, selects a right degree of precision, remove outliers and generally filter out irrelevant or non representative measurements. Techniques exist for preprocessing data in data mining, however, they often assume that data is available in its entirety and that sufficient computing power and memory space is available. As will be seen in more details in the sections devoted to data streaming, algorithms have to be completely reconsidered in the limited rationality and lifelong setting. At the other end of the learning process, the results of learning that have to be stored or shared between agents should require only compact transmissions and storage.

Overall, new techniques for intelligent sampling, to use generate synopsis and summarized information, and to output approximate solutions are required. But this brings a new problem. Since the degree of approximation is dependent on the specifics of the ubiquitous system and on the application, ubiquitous data mining systems must be able to adapt to the characteristics of the problem and to their own limitations. In other words, they must be *self-aware* to a certain degree in order to optimize their own parameters and deliver on-time the best possible analysis. In the field of data mining per se, a powerful meta-learning approach has been developed since the 90s, that of *ensemble learning* [31]. The technique makes use of the combination of multiple models learned by *weak learners* in order to **boost** the overall performance of the learning system. In the context of ubiquitous data mining, it is tempting to view these ensemble learning techniques as means to turn the curse of distributed and limited data mining algorithms into a strength, by using local models as base learners. Nevertheless, so far, few works [27] have addressed the problem of incremental, online, and distributed maintenance of ensembles of decision models [20].

2.2.1 A lifelong perspective for Machine Learning

Aside the computational and communication problems inherent in ubiquitous environments, there are issues linked to the nature of the data themselves. As soon as one is contemplating lifelong learning tasks with data originating from various places, new problems arise. Mostly, they resort to two categories of questions. The first one is related to the fact that data are geographically situated. The second is related to the temporal nature of the data generation of which several aspects are important for learning. Namely, data now arrive in streams and can only be processed on a one-pass basis, data is no longer i.i.d. and the underlying regularities may change over time.

In the following, these two categories of problems are examined in turn.

Mining spatially tagged data. There are now several data generation environments where the nodes generating the data are spatially spread and interrelated and where these relations are meaningful and important for data mining tasks. New types of applications are thus emerging in health services, in environmental monitoring or in distributed and real-time control of vehicles. It is also

possible to record characteristics of the landscape itself as in satellite recordings and to monitor the spatial location of objects. The latter has become very popular with the emergence of mobile phones, GPS and RFID. In sensor networks, for instance, sensors are often explicitly spatially related to each other and this spatial information can be considered as a king of meta tagging of the data. In other applications, where, for instance, the agents themselves are moving (e.g. vehicles on roads), GPS information can accompany the set of measurements made.

In all of these applications, the spatial information is essential in the processing of the data and the discovery of meaningful regularities. So far, works on spatial data mining has been scarce even though applications in satellite remote sensing and geographical databases have spurred a growing interest in methods and techniques to augment databases management and mining with spatial reasoning capabilities.

The time situation of data. The fact that, in ubiquitous environments, data are produced on a real-time basis, or, at least, in a sequential fashion, and that the environment and the task at hand may change over time, profoundly modifies the underlying assumptions on which rest most of the existing learning techniques and demands the development of new principles and new algorithms.

In the following, we discuss these new set of issues along two main directions. The first one deals with the sequential nature of data, the fact that it comes as streams of indefinite length and must be processed with limited resources. The second ones discusses the consequences of the fact that data can no longer be considered as independently and identically distributed.

2.3 Distributed Data Mining

The combination of distributed systems and the pervasive/ubiquitous computing paradigms can lead into the area of distributed data mining. It is a broad area characterized by combinations of multiple learning algorithms as well as multiple learning problems. These learning problems are also directed by constraints such as security, privacy, communication costs and energy restrictions.

Autonomous Agents

The idea of ubiquitous knowledge discovery and autonomous agents is closely related. An agent need to be able to discover, sense, learn, and act on its own. The research on agents is extensive. There are basically two types of agent architectures, one in which the agents are tied to one node and one in which the agents move between nodes. In systems such as Objectspace's *Voyager*, the agents move between the machines, while a peer-to-peer system such as *JXTA* only allows agents to be tied to the nodes. In both systems the agents either reside or migrate to nodes that are prepared for receiving. Distributed objects can be seen as a premise for agents: only some additional characteristics need to be adorned to the distributed object. It has been noted that agents normally differ from machine learning algorithms in that the nature of the problems addressed by the agents are larger than those traditionally associated with machine learning [11]. Intelligent agents are more involved software than just an agent, which

can be compared to a distributed object. The term distributed object perhaps belongs to systems in which the focus is more on the distributed system than on the interaction between the agents. One such example is the *Java Agents for Meta-Learning* (JAM) system developed by Stolfo et al. [32]. The agent can communicate with other agents using the *Knowledge Query and Manipulation Language* (KQML) [13] together with the Knowledge Interchange Formalism (KVI) [17] agent information exchange format. These communications and associated exchange formats allow the agent to express first order requests. A complementary system for agent corporation is the *Open Agent Architecture* (OOA) [24]. In some cases the agents will query other agents and perhaps exchange examples or models. To exchange an example sounds simple, but it requires that the data format is agreed to and understood on both sides of the exchange. Research in general purpose languages for model exange is recent. An example is the Predictive Model Markup Language (PMML), a XML mark up language to describe statistical and data mining models.

The strong limitations of the first scenario is discussed in [28]. The authors point out *'a mismatch between the architecture of most off-the-shelf data mining algorithms and the needs of mining systems for distributed applications'*. Such mismatch may cause a bottleneck in many emerging applications. Some hardware limitations related to the limited bandwidth channels. Most important, in applications like monitoring, centralized solutions introduce delays in event detection and reaction, that can make mining systems useless.

2.3.1 Sensor Networks

Restrictions like size and power source will define how complex is the processing performed in the sensors. We may have in one extreme centralized systems and in the other extreme completely distributed systems, where almost all the intelligent processing is performed in the sensors. The solution considered in Ubuiquitous Data Mining can broadly be seen as belonging to one of the following categories: data-centric, process-based or network-based:

Categories for Ubiquitous Data Mining

- Data-centric solutions
  The data is preprocessed before the actual learning phase takes place. It could be in the form of processing a subset of data such as sampling and load shedding. It also could be in the form of dimensionality reduction such as sketching. Finally it could be by summarizing the streaming data such as aggregation and creating data synopsis [2] .

- Process-based solutions
  The focus of this category of approaches is to deal with the actual processing of the algorithm. Approximation and randomized algorithms are the traditional solutions used in this category [26]. Recently, sliding windows is used in a two-fold objective. The first is to capture the most recent output. The other objective is solve the problem of concept drift.

- Network-based solutions
  Different networking approaches can provide approximate results taking

into consideration the limitations of ubiquitous computing environments. Peer-to-Peer (P2P) computing [10], clustering of sensor nodes and efficient routing [35], and grid computing [7] are typical approaches used in this category. In P2P computing, the data mining algorithm is executed locally onboard a sensor node or a handheld device. The algorithm allows message exchange in order to integrate local results. Approximate global model could be computed accordingly. Clustering of sensor nodes aims at prolonging the network life-time by choosing nodes with highest energy to be the cluster heads. Cluster heads then collect and aggregate data to be sent to the base station. In grid computing, a so-called knowledge grid is built to benefit from the services provided by the grid in the distributed data mining process. These services include information and resource management, communication and authentication.

Sensor networks can be used to detect the occurrence of complex events patterns with spatial and temporal characteristics. Traditionally, sensor networks communicate sensed data to a central server that runs offline data mining algorithms. In the KDUbiq perspective, analysis should be done done *in situ* using local information. The main difference is the conceptual framework. Current Data Mining techniques assume iid examples, static models, finite training sets, unrestricted resources. Even ignoring resources, they might be applied with high maintenance costs: retraining models from time to time. KDubiq approach (dynamic models that evolve over time, incorporating change detection mechanisms, taking into account the resources available) is the most appropriate option.

An Illustrative Example: Distributed Clustering. The scenario is in a sensor network where each sensor produce a continuous stream of data. Suppose we have m distributed sites, and each site i has a data source S_i^t at time t. The goal is to continuously maintain a k-means clustering of the points in $S^t = \cup_i^m S_i^t$.

The distributed algorithm presented in *Conquering the Divide: Continuous Clustering of Distributed Data Streams* [9] is based on the *Furthest Point* clustering. The base idea consists of selecting randomly the first cluster center c_1 among data points. Subsequent $k-1$ cluster centers are chosen as the points that are more distant from the previous centers c_1, c_2, ..., c_{i-1}, by maximizing the minimum distance to the centers. This algorithm requires k passes over training points. It has an interesting property. It ensures a 2-approximation of the optimal clustering. A skeleton of the proof is: Suppose a $k+1$ iteration, producing $k+1$ points separated by a distance at least D. The optimal k clustering must have a diameter at least D. By the triangular inequality the chosen clustering has diameter at most 2D. Based on the *Furthest Point* algorithm the authors developed a one pass clustering algorithm: the *Parallel Guessing Clustering*. The base idea consists of picking an arbitrary point as the first center, and for each incoming point p compute $r_p = min_{c \in C} d(p, c)$. If $r_p > R$ Set $C = C \cup p$. This strategy would be Ok, if we know R, the problem is that R is unknown in advance! The solution proposed in [9] consists of making multiple

guesses for R as $(1+\epsilon/2)$, $(1+\epsilon/2)^2$, $(1+\epsilon/2)^3$, and run the algorithm in parallel. If a guess R generates more than k centers, implies R is smaller than the optimal radius (R_o). For a guess of $R \geq 2R_o$ that find k or fewer centers, and generate a valid clustering.

Each local site maintains a *Parallel Guessing Algorithm* using its own data source. Whenever it reachs a solution, it sends to the coordinator the k centers and the radius R_i. Each local only re-send information when the centers change. The coordinator site maintains a *Furthest Point Algorithm* over the centers sent by local sites.

Data Streams

Data stream mining techniques use the above approaches in different data mining strategies including clustering, classification, frequent pattern discovery, time series analysis and change detection [15]. While few of these techniques are resource-aware when applying the above solution approaches, others are not. This problem needs to be addressed in order to realize UDM algorithms in real-life applications. For example if a data stream mining technique uses an approximate solution to preserve the available memory with constant factor, the algorithm may run out of memory due to the extremely low availability of it in a ubiquitous computing environment. However, if the algorithm is resource-aware, the approximation factor can change over time in order to cope with the critically low availability of memory. The same analogy applies to the different available resources. This has been demonstrated in [14]. The area of adaptation and resource-awareness is still open. Many data stream mining algorithms are required to be resource-aware and adaptive in order to realize its applicability in ubiquitous computing environments.

2.4 Knowledge Discovery from Data Streams

In the last two decades, machine learning research and practice has focused on batch learning usually with small datasets. In batch learning, the whole training data is available to the algorithm, that outputs a decision model after processing the data eventually (or most of the times) multiple times. The rationale behind this practice is that examples are generated at random accordingly to some stationary probability distribution. Most learners use a greedy, hill-climbing search in the space of models.

2.4.1 Static versus Streaming

What distinguishes current data sets from earlier ones are the continuous flow of data and the automatic high-speed data feeds. We do not just have people who are entering information into a computer. Instead, we have computers entering data into each other [26]. Nowadays there are applications in which the data is best modeled not as persistent tables but rather as transient data streams. In some applications it is not feasible to load the arriving data into a traditional DataBase Management Systems (DBMS), and traditional DBMS are not designed to directly support the continuous queries required in these applications [2].

Streaming Algorithms

Algorithms that process data streams deliver approximate solutions, providing a fast answer using few memory resources. They relax the requirement of an exact answer to an approximate answer within a small error range with high probability. In general, as the range of the error decreases the space of computational resources goes up. In some applications, mostly database oriented, an approximate answer should be within an admissible error margin. Some results on tail inequalities provided by statistics are useful to accomplish this goal. The basic general bounds on the tail probability of a random variable (that is, the probability that a random variable deviates greatly from its expectation) include the Markov, Chebyshev and Chernoff inequalities [25].

Data Streams Management Systems developed a set of techniques that store compact stream summaries enough to approximately solve queries. All these approaches require a trade-off between accuracy and the amount of memory used to store the summaries, with an additional constrain of small time to process data items [26]. The most common problems end up to compute quantiles, frequent item sets, and to store frequent counts along with error bounds on their true frequency.

In the streaming model (see [26]), the input elements $a_1, a_2, \ldots, a_j, \ldots$ arrive sequentially, item by item and describe an underlying function A. Streaming models differ on how a_i describe A. We can distinguish between:

1. Insert Only Model: once an element a_i is seen, it can not be changed.

2. Insert-Delete Model: elements a_i can be deleted or updated

From the viewpoint of a data streams management system, several research issues emerge. For instance, one such issue is related to the need of approximate query processing techniques in order to evaluate queries that require unbounded amount of memory. Sampling techniques have been used to handle situations where the flow rate of the input stream is faster than the query processor. One important question is raised by the existence of of blocking operators (e.g. aggregation and sorting) in the presence of unending streams. It is essential to identify them and to find ways to circumvent their blocking effect. The following table summarizes the main differences between traditional and stream data processing:

	Traditional	Stream
Number of passes	Multiple	Single
Processing Time	Unlimited	Restricted
Available Memory	Unlimited	Fixed
Result	Accurate	Approximate
Distributed	No	Yes

2.4.2 When data points are no longer i.i.d.

When the samples of data are both spatially and time situated, data points can no longer be considered as independently and identically distributed, a fact that

is compounded when the underlying generative process is itself changing over time.

One consequence is that the statistical theory of learning [33] does not hold anymore. In particular, the inductive criteria that are based on additive measures of cost (e.g. the empirical risk and the real risk associated with a candidate hypothesis) are no longer satisfactory. The question then becomes: how to replace this fundamental theory? Works on weak long term correlations exist in statistics, but they are still far from answering the challenge that faces machine learning in this respect.

Ordering Effects

One interesting question concerns ordering effects: the fact that the result of a (on-line) learning session may depend on the order of presentation of the data points. This has usually been considered as a nuisance that one should try to get rid of. However, the order of the data points can clearly convey potentially useful information about the *evolution* of the underlying generative process, or even, this order could result from a clever and helpful teacher. In this case, one should on the contrary try to take advantage from this source of information. Apart from a few works in constructive induction and some pioneering works in inductive logic programming, almost everything remains to be done in this direction. Reasoning about the evolution of the learning process is also a potential research line that remains almost unexplored.

Concretely, data streams require that the learners be endowed with on-line or incremental learning capabilities. This means in particular that they should be able to process the data sequentially as they arrive and that they should be able to produce either decisions or hypotheses about the surrounding world in an anytime fashion. Since, their memory is limited and because the underlying generative process, or the task at hand, may change, they have to be able to measure the relevance of a piece of information over time, and be able to *forget* no longer relevant data pieces.

Questions about how to sample and summarize the data, how to carry on on-line learning are discussed below.

2.4.3 Where we Are

Recent developments in Machine Learning point out directions for learning in ubiquitous environments. For example, in [18] the authors present a general method to learn from arbitrarily large databases. The method consists of deriving an upper bound for the learner's loss as a function of the number of examples used in each step of the algorithm. Then use this to minimize each step's number of examples, while guaranteeing that the model produced does not differ significantly from the one that would be obtained with infinite data. This general methodology has been successfully applied in k-means clustering [18], decision trees [12, 19, 16], etc. The proposed method can be useful to solve one aspect of Learning in ubiquitous environments. Nevertheless many others remain unsolved. For example, learning from distributed data, we need efficient methods in minimizing the communication overheads between nodes. Work in this direction appears in [3].

illustrative Algorithm. We have pointed out the necessity of new algorithms for clustering variables in the stream setting. The Online Divisive-Agglomerative Clustering (ODAC) system [30] continuously maintains a tree-like hierarchy of clusters that evolves with data. ODAC uses a top-down strategy. The splitting criterion is a correlation-based dissimilarity measure among time series, splitting each node by the farthest pair of streams, which defines the diameter of the cluster. In stationary environments expanding the structure leads to a decrease in the diameters of the clusters. The system uses a merge operator, that agglomerates two sibling clusters, in order to react to changes in the correlation structure between time series. The splitting and merge operators are triggered in response to changes in the diameters of existing clusters. The system is designed to process thousands of data streams that flow at high-rate. The main features of the system include update time and memory consumption that do not depend on the number of examples in the stream. Moreover, the time and memory required to process an example decreases whenever the cluster structure expands.

Approximation and Randomization techniques has been used to solve distributed streaming learning problems. Approximation allows answers that are correct within some fraction ϵ of error, while *randomization* allows a probability δ of failure. The base idea consists of mapping a very large input space to a small synopsis of size $O(\frac{1}{\epsilon^2} log(\frac{1}{\delta}))$. Approximation and Randomization techniques has been used to solve problems like measuring the entropy of a stream, association rule mining frequent items [23], k-means clustering for distributed data streams using only local information [9], etc.

illustrative Algorithm. Hash functions as been used to project attributes with huge domains into lower space dimensions. The Count-Min Sketch has been used to approximately solve point queries, range queries, and inner product queries. Here we present a simple couting example: How many times an item in high-speed stream as been seen so far.

A Count-Min Sketch is an array of $w \times d$ in size. Given a desired probabilty level (δ), and an admissible error (ϵ), $w = 2/\epsilon$ and $d = log(1/\delta)$. Each entry x in the stream, is mapped to one bucket per row. It uses d hash functions to map vector entries to $[1..w]$. At any time, the estimate $\hat{x}[j]$ is given by taking $min_k CM[k, h_k(j)]$. The important properties of this estimate are: $x[j] \leq \hat{x}[j]$ and $\hat{x}_i \leq \epsilon \times ||x_i||_1$, with probability $1 - \delta$.

The idea of the agent's *limited rationality* lead us to ensemble models used to collectively solve problems. In [20], the authors proposed a method that offer an effective way to construct a redundancy-free, accurate, and meaningful representation of large decision-tree ensembles often created by popular techniques such as Bagging, Boosting, Random Forests and many distributed and data stream mining algorithms.

2.4.4 Online, Anytime and Real-time Learning

The challenge problem for data mining is the ability to continuously maintain an accurate decision model. This requires learning algorithms that can modify the current model whenever new data is available at the rate of data arrival. Moreover, they should forget older information when data is outdated. In this context, the assumption that examples are generated at random according to a stationary probability distribution does not hold, at least in complex systems and for large periods of time.

In the presence of a non-stationary distribution, the learning system must incorporate some form of forgetting past and outdated information. Learning from data streams require incremental learning algorithms that take into account these changes in the data generating process. Solutions to these problems require new sampling and randomization techniques, and new approximate, incremental and decremental algorithms. In [18], the authors identify desirable properties of learning systems that are able to mine continuous, high-volume, open-ended data streams as they arrive. Learning systems should be able to process examples and answering queries at the rate they arrive. Overall, some desirable properties for learning in data streams include: incrementality, online learning, constant time to process each example, single scan over the training set, and taking drift into account.

Cost-Performance Management Incremental learning is one fundamental aspect for the process of continuously adaptation of the decision model. The ability to update the decision model whenever new information is available is an important property, but it is not enough. Another required operator is the ability to *forget* past information [22]. Some data stream models allow delete and update operators. Sliding windows models require forgetting old information. In all these situations the incremental property is not enough. Learning algorithms need forgetting operators that reverse learning: decremental unlearning [8].

The incremental and decremental issues requires a continuous maintenance and updating of the decision model as new data is available. Of course, there is a trade-off between the cost of update and the gain in performance we may obtain. The update of model depends of the complexity of its representation language. For instance, very simple models, using few free-parameters, are easy to adapt in face of changes. Small and potentially insignificant variations in the data will not result in unwarranted changes in the learned regularities. The variance of the hypothesis class is said to be limited. However, the downside is an associated high bias. The hypothesis space cannot accommodate a large variety of data dependencies. This is known as the bias-variance tradeoff. This tradeoff is fundamental in classical one-shot learning. It becomes even more so in on-line learning. In addition, the more complex the hypothesis space (the larger the number of effective parameters), the more costly are the update operations. Effective means of controlling it must be devised. Continuous learning therefore requires efficient control strategies in order to optimize the trade-off between the gain in performance and the cost of updating. To this aim, incremental and

decremental operators have been proposed, including sliding windows methods.

An illustrative example is the case of ensemble learning techniques, such as boosting, that rely on learning multiple models. Theoretical results show that it is thus possible to obtain arbitrary low errors by increasing the number of models. To achieve a linear reduction of the error, we need an exponential increase in the number of models. Finding the *Breakeven* point, that is the point where costs equalize benefits, and not going behind that, is a main challenge.

Monitoring Learning When data flows over time, and, at least, in case of large periods of time, the assumption that the examples are generated at random according to a stationary probability distribution becomes highly unlikely. In complex systems and for large time periods, we should expect changes in the distribution of the examples. A natural approach for these *incremental tasks* are *adaptive learning algorithms*, incremental learning algorithms that take into account concept drift.

Concept drift means that the concept related to the data being collected may shift from time to time, each time after some minimum permanence. Changes occur over time. The evidence for changes in a concept are reflected in some way in the training examples. Old observations, that reflect the past behavior of the nature, become irrelevant to the current state of the phenomena under observation and the learning agent must forget that information.

The nature of change is diverse. Changes may occur in the context of learning, due to changes in hidden variables, or in the characteristic properties of the observed variables. In fact, it is usual to distinguish between:

- Changes in the underlying distribution over the instance descriptions (denoted $D_{\mathcal{X}}$, the distribution over the description space $\mathcal{X}$).
- Changes in the conditional distribution of the label w.r.t. the description (denoted $D_{\mathcal{Y}|\mathcal{X}}$, where $\mathcal{Y}$ is the label space). This is usually called "concept drift".
- Changes in both of the distributions.

In the former case, the change affects the space of the available instances, but not the underlying regularity. This, however, may require changes in the decision rules in order to optimize the decision process over the dense regions of the instance space. In the second case, the identified decision rule has to be adapted.

Most learning algorithms use blind methods that adapt the decision model at regular intervals without considering whether changes have really occurred. Much more interesting is explicit change detection mechanisms. The advantage is that they can provide meaningful description (indicating change-points or small time-windows where the change occurs) and quantification of the changes. The main research issue is how to incorporate change detection mechanisms in the learning algorithm. Embedding change detection methods in the learning algorithm is a requirement in the context of continuous flow of data. The level of *granularity* of decision models is a relevant property, because if can allow

partial, fast and efficient updates in the decision model instead of rebuilding a complete new model whenever a change is detected. The ability to recognize seasonal and re-occurring patterns is an open issue.

Novelty Detection. Novelty Detection refers to the automatic identification of unforeseen phenomena embedded in a large amount of normal data. It corresponds to the appearance of a new concept (e.g. a new label or a new cluster) *from unlabelled data*. Learning algorithms must then be able to identify and learn new concepts. *Novelty* is always a relative concept with regard to our current knowledge. Intelligent agents that act in dynamic environments must be able to learn conceptual representations of such environments. Those conceptual descriptions of the world are always incomplete. They correspond to what it is *known* about the world. This is the *open* world assumption as opposed to the traditional *closed* world assumption, where what is to be learnt is defined in advance. In open worlds, learning systems should be able to extend their representation by learning new concepts from the observations that do not match the current representation of the world. This is a difficult task. It requires to identify the *unknown*, that is, the limits of the current model. In that sense, the *unknown* corresponds to an *emerging pattern* that is different from *noise*, or *drift* in previously known concepts.

2.4.5 Issues and Challenges in Learning from Distributed Data Streams

Streaming data and domains offer a nice opportunity for a symbiosis between Streaming Data Management Systems and Machine Learning. The techniques developed to estimate synopsis and sketches require counts over very high dimensions both in the number of examples and in the domain of the variables. The techniques developed in data streams management systems can provide tools for designing Machine Learning algorithms in these domains. On the other hand, Machine Learning provides compact descriptions of the data than can be useful for answering queries in DSMS.

Incremental Learning and Forgetting. In most applications, we are interested in maintaining a decision model consistent with the current status of the nature. This lead us to the sliding window models where data is continuously inserted and deleted from a window. Learning algorithms must have operators for incremental learning and forgetting. Incremental learning and forgetting are well defined in the context of predictive learning. The meaning or the semantics in other learning paradigms (like clustering) are not so well understood, very few works address this issue.

Change Detection. *Concept drift* in the predictive classification setting is a well studied topic. In other learning scenarios, like clustering, very few works address the problem. The main research issue is how to incorporate change detection mechanisms in the learning algorithm for different paradigms.

Feature Selection and Pre-processing. Selection of relevant and informative features, discretization, noise and rare events detection are common tasks in Machine Learning and Data Mining. They are used in a one-shot process. In the streaming context the semantics of these tasks changes drastically. Consider the feature selection problem. In streaming data the concept of *irrelevant* or *redundant* features are now restricted to a certain period of time. Features previously considered *irrelevant* may become *relevant*, and vice-versa to reflect the dynamics of the process generating data. While in standard data mining, an irrelevant feature could be ignored forever, in the streaming setting we need still monitor the evolution of those features. Recent work based on the *fractal dimension* [5] could point interesting directions for research.

Ubiquity in the Feature Space. In the static case, similar data can be described with different schemata. In the case of dynamic streams, the schema of the stream can also change. We need algorithms that can deal with evolving feature spaces over streams. There is very little work in this area, mainly pertaining to document streams. For example, in sensor networks, the number of sensors is variable (usually increasing) over time. For instance, clustering of data coming through streams on different sensing and computing sites is a growing field of research that brings completely new algorithms (see for instance [21, 1]).

Evaluation Methods and Metrics. An important aspect of any learning algorithm is the hypothesis evaluation criteria. Most of evaluation methods and metrics were designed for the static case and provide a single measurement about the quality of the hypothesis. In the streaming context, we are much more interested in how the evaluation metric evolves over time. Results from the *sequential statistics* [34] may be much more appropriate.

There is a fundamental difference between learning from small datasets and large datasets. As pointed-out by some researchers [6], current learning algorithms emphasize variance reduction. However, learning from large datasets may be more effective when using algorithms that place greater emphasis on bias management.

2.5 Where We Want to Go: Emerging Challenges and Future Issues

KDUbiq points out to systems and algorithms with high level of autonomy. These systems address the problems of data processing, modeling, prediction, clustering, and control in changing and evolving environments. They self-evolve their structure and knowledge on the environment.

In contrast to the main stream in Machine Learning theory and practice, **KDubiq points out a world in movement where things evolve in time and space**. The main difference is the conceptual framework. Current DM

techniques assume a static world, that we can monitor using unrestricted resources. If we ignore resource constraints, they might be applied with high maintenance costs (retraining models from time to time). In the KDubiq perspective the world is in movement; any learning task requires decision models that evolve over time, incorporating change detection mechanisms, and taking into account the resources available. The relevant point *is not what we can do more, but what we can better*.

The definition of *standards* to represent and exchange models, *incorporation of domain knowledge*, *vizualization*, the *definition of processes and systemic approaches*, are traditional issues in data mining that are reinforced from the emerging applications like *bioinformatics*, *semantic Web*, *sensor networks*, *radio frequency identification*, etc.

The main lesson we, researchers in knowledge discovery, can learn from the challenges that ubiquity poses is that learning algorithms are limited. Real world is much greater even than a network of computers. The design of learning algorithms must take memory, space, time, communication, etc into account.

Simple objects that surround us are changing from static, inanimate objects into adaptive, reactive systems with the potential to become more and more useful and efficient. Smart things associated with all sort of networks offers new unknown possibilities for the development and self-organization of communities of intelligent communicating appliances. Learning in these contexts must be *non pervasive*, and become invisible.

References

[1] Charu Aggarwal, editor. *Data Streams – Models and Algorithms*. Springer, 2007.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In Phokion G. Kolaitis, editor, *Proceedings of the 21nd Symposium on Principles of Database Systems*, pages 1–16. ACM Press, 2002.

[3] Amir Bar-Or, Daniel Keren, Assaf Schuster, and Ran Wolff. Hierarchical decision tree induction in distributed genomic databases. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1138–1151, 2005.

[4] D. Barbara. Requirements for clustering data streams. *SIGKDD Explorations*, 3(2):23–27, January 2002.

[5] D. Barbara and P. Chen. Using the fractal dimension to cluster datasets. In *Proc. of the 6th International Conference on Knowledge Discovery and Data Mining*, pages 260–264. ACM Press, 2000.

[6] D. Brain and G. Webb. The need for low bias algorithms in classification learning from large data sets. In T.Elomaa, H.Mannila, and H.Toivonen,

editors, *Principles of Data Mining and Knowledge Discovery PKDD-02*, pages 62–73. LNAI 2431, Springer Verlag, 2002.

[7] M. Cannataro, D. Talia, and P. Trunfio. Distributed data mining on the grid. *Future Generation Computer Systems*, 18(8):1101–1112, 2002.

[8] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Proceedings of the 13th Neural Information Processing Systems*, 2000.

[9] Graham Cormode, S. Muthukrishnan, and Wei Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *ICDE*, pages 1036–1045, 2007.

[10] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing special issue on Distributed Data Mining*, 10(4):18–26, 2006.

[11] W. Davies and P. Edwards. Agent-Based Knowledge Discovery. In *AAAI Spring Symposium on Information Gathering*, 1995.

[12] P. Domingos and G. Hulten. Mining High-Speed Data Streams. In Ismail Parsa, Raghu Ramakrishnan, and Sal Stolfo, editors, *Proceedings of the ACM Sixth International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM Press, 2000.

[13] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press.

[14] M. Gaber, M and Yu P. S. A framework for resource-aware knowledge discovery in data streams: a holistic approach with its application to clustering. In *ACM Symposium Applied Computing*, pages 649–656. ACM Press, 2006.

[15] J. Gama and M. Gaber (Eds). *Learning from Data Streams – Processing techniques in Sensor Networks*. Springer, 2007.

[16] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 523–528, New York, NY, USA, 2003. ACM.

[17] M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Stanford University, Stanford, CA, USA, 1992.

[18] G. Hulten and P. Domingos. Catching up with the data: research issues in mining data streams. In *Proc. of Workshop on Research issues in Data Mining and Knowledge Discovery*, 2001.

[19] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International conference on Knowledge discovery and data mining*, pages 97–106. ACM Press, 2001.

[20] H. Kargupta and H. Dutta. Orthogonal Decision Trees. In *Proceedings of The Fourth IEEE International Conference on Data Mining (ICDM'04)*, Brighton, UK, November 2004.

[21] H. Kargupta and K. Sivakumar. *Data Mining: Next Generation Challenges and Future Directions*, chapter Existential Pleasures of Distributed Data Mining. AAAI/MIT Press, 2004.

[22] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *VLDB 04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages 180–191. Morgan Kaufmann Publishers Inc., 2004.

[23] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 346–357. Morgan Kaufmann, 2002.

[24] D. Martin, A. Cheyer, and D. Moran. The Open Agent Architecture: a framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1/2):91–128, 1999.

[25] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[26] S. Muthukrishnan. *Data streams: algorithms and applications*. Now Publishers, 2005.

[27] N. Oza. *Online Ensemble Learning*. PhD thesis, University of California, Berkeley, 2001.

[28] B. Park and H. Kargupta. *Data Mining Handbook*, chapter Distributed Data Mining: Algorithms, Systems, and Applications. Lawrence Erlbaum Associates, 2002.

[29] G. Riva, F. Vatalaro, F. Davide, and M. Alcaiz, editors. *Ambient Intelligence – The Evolution of Technology, Communication and Cognition Towards the Future of Human-Computer Interaction*, volume 6. OCSL Press, January 2005.

[30] P. P. Rodrigues, J. Gama, and J. Pedroso. Hierarchical clustering of time series data streams. *IEEE Transactions on Knowledge and Data Engineering*, (to appear) 2008.

[31] R. Schapire. Strength of weak learnability. *Journal of Machine Learning*, 5:197–227, 1990.

[32] Salvatore J. Stolfo, Andreas L. Prodromidis, Shelley Tselepis, Wenke Lee, Dave W. Fan, and Philip K. Chan. JAM: Java agents for meta-learning over distributed databases. In *Knowledge Discovery and Data Mining*, pages 74–81, 1997.

[33] V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, 1995.

[34] A. Wald. *Sequential Analysis*. John Wiley and Sons, Inc., 1947.

[35] O. Younis and S. Fahmy. Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, 2004.