

On-line learning: where are we so far?

Antoine Cornuéjols
AgroParisTech
UMR 518, Dept. MMIP
16, rue Claude-Bernard
F-75231 Paris cedex 05, France

September 24, 2009

1 Introduction

While Machine Learning is still a young field, having approximately 50 years of existence, its history is now sufficiently long that some historical perspective and deep trends can be perceived. Thus, the first learning algorithms were all incremental. For instance, the perceptron, the Checker program, ARCH or the Candidate Elimination Algorithm for Version Space learning, to name but a few. Various reasons motivated this state of affair. One was that these programs were, in part at least, aimed at simulating human learning, which is mostly incremental in nature. Another was that the very limited available computing power, by today's standard, prevented the storage and processing of large data bases of learning examples. However, this rule was completely overturned in the 80s. A wealth of new algorithms were developed: viz. decision trees, feed-forward neural networks, Support Vector Machines, Grammatical inference systems, and many more. Almost all are “batch” learners, meaning that they learn from a single batch of examples, optimizing some inductive criterion over the *whole* training set. If new training instances are made available, then the learning process must start all over again from scratch.

It is interesting to examine reasons for this complete about-turn from the previous period. Thus, section 2 in particular provides the fundamentals of the now standard setting. Recent years, however, have witnessed a renewal of interest for on-line learning. Sections 2.1 provides reasons for this and the issues that are raised in consequence. Then, section 3, describes essential issues in on-line learning. Each of these issues remains essentially to solve, both at a theoretical level but also at the engineering level of conceiving new learning algorithms. Section 4, describes a special on-line learning framework called *tracking*. We show how new ideas could be brought to play in order to provide both original theoretical tools and learning methods to solve this problem. We think this nicely underlines the range of issues at play as well as the type of new ideas that we could call upon. Finally, we conclude with an appeal to a new scientific outlook for learning.

2 The standard setting: one-shot and i.i.d.

An agent learns when it interacts with the world, using percepts to make decisions and take actions, and then measuring its performance, without which it would not be able

to sense in what direction it should modify its decision making process. In order to learn, one has to be able to compare situations, that is to measure similarities or to make generalizations. One central concern in the study of learning has focussed on generalization and on questions such as: which conditions allow one to generalize? how to perform generalization? how to evaluate the confidence in the result of generalization?

Most studies in inductive learning assume that the learning agent comes across random feature vectors \mathbf{x} (called the “observables”), which are generated according to the following two-stage process. First, a random class e.g. $y \in \{-1, 1\}$ is selected using the *a priori* probabilities \mathbf{p}_y ; then, the observed feature vector \mathbf{x} is generated according to the class-conditional distribution $\mathbf{p}_{\mathcal{X}|y}$. The distribution over labelled patterns is thus given by $\mathbf{p}_{\mathcal{X}\mathcal{Y}} = \mathbf{p}_y \mathbf{p}_{\mathcal{X}|y} = \mathbf{p}_{\mathcal{X}} \mathbf{p}_{y|\mathcal{X}}$.

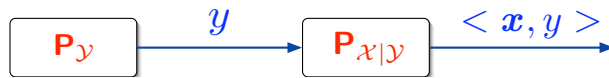


Figure 1: The two-stage generation of learning examples.

When acting as a *classifier*, the agent is facing the following problem: given a realization of the measured feature vector \mathbf{x} , decide whether the unknown object engendering \mathbf{x} belongs to class -1 or 1. A classifier or decision rule, in this setting, is simply a map $h : \mathcal{X} \rightarrow \{-1, 1\}$, which determines the class $h(\mathbf{x})$ to which an observed feature vector \mathbf{x} should be assigned. In the context of Machine Learning, this map is called a *hypothesis*, hence the notation h^1 . It is thus possible to define the performance of a classifier (or hypothesis) as the *probability of error* given by:

$$L(h) = \mathbf{p}_{\mathcal{X}\mathcal{Y}}\{h(\mathbf{x}) \neq y\} \quad (1)$$

More generally, if different costs are assigned to different types of errors², specified through the definition of a *loss function* ℓ , defined as:

$$\begin{aligned} \ell(h) : \mathcal{X} \times \mathcal{Y} &\rightarrow \mathbb{R}_+ \\ (\mathbf{x}, y) &\mapsto \ell(h(\mathbf{x}), y) \end{aligned} \quad (2)$$

Then, the performance of a classifier is defined as a *risk*, which is an expectation over the possible events:

$$R(h) = \mathbb{E}[\ell(h(\mathbf{x}), y)] = \int_{\mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}} \ell(h(\mathbf{x}), y) \mathbf{p}_{\mathcal{X}\mathcal{Y}} d(\mathbf{x}, y) \quad (3)$$

If the *a priori* probabilities \mathbf{p}_y and conditional distributions $\mathbf{p}_{y|\mathcal{X}}$ are known, the optimal decision rule, in the sense of minimum probability of error (or of minimum risk) is the *Bayes decision rule*, denoted h^* and defined as:

$$h^*(\mathbf{x}) = \underset{y \in \{0,1\}}{\text{ArgMin}}(\ell(y, 1 - y) \mathbf{P}\{y|\mathbf{x}\}) \quad (4)$$

In many situations, however, these distributions are unknown or only partially known, but one is given a training set $\mathcal{S}_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$

¹ In Statistics, this notion is known as a *model*.

² In medicine, for instance, it is much more costly to miss an appendicitis diagnosis, than to decide to operate, only to discover it was a false alert.

supposed to be drawn according to the unknown probability distribution $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$. The basic assumption enabling learning is that all the data (both observed and unseen) are generated by the same process, which is formalized by saying that the data is sampled independently from a fixed identical probability distribution (*i.i.d. sampling*).

The **learning problem** is thus: given a training set consisting of labelled objects supposedly drawn *i.i.d.* from the unknown distribution $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$, find a function h that assigns labels to objects such that, if new objects are given, this function will label them correctly.

Short of attaining a perfect identification of the target dependency between the feature vectors and their label, the performance of a classifier or hypothesis is measured with the risk $R(h)$ (see equation 3). A large part of the theory in Machine Learning focuses on finding conditions for constructing good classifiers h whose risk is as close to $R^* = R(h^*)$ as possible.

A natural and simple approach is to consider a class \mathcal{H} of hypotheses $h : \mathcal{X} \rightarrow \{-1, 1\}$ and to estimate the performance of the hypotheses based on their empirical performance measured on the learning set. The most obvious choice to estimate the risk associated with a hypothesis is to measure its *empirical risk* on the learning set \mathcal{S}_m :

$$R_m(h) = \frac{1}{m} \sum_{i=1}^m \ell(h(\mathbf{x}_i), y_i) \quad (5)$$

which, in the case of binary classification with 0-1 loss, gives:

$$R_m(h) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}_{(h(\mathbf{x}_i) \neq y_i)}, \quad (6)$$

where one counts the number of prediction errors on the training set.

In this framework, it is then natural to select a hypothesis with the minimal empirical risk as a most promising one to classify unseen events. This inductive criterion is called the *Empirical Risk Minimization* principle. According to it, the best candidate hypothesis is:

$$\hat{h}^* = \underset{h \in \mathcal{H}}{\text{ArgMin}} R_m(h) \quad (7)$$

However, the statistical theory of learning has shown that it is crucial that the hypothesis space from which candidate hypotheses are drawn be limited in terms of its expressive power. Specifically, one should not concentrate only on finding hypotheses that minimize the empirical risk irrespective of the hypothesis space, but one should indeed take into account its *capacity* or expressive power. In fact, the less diverse is \mathcal{H} , the tighter the link between the measured empirical risk $R_m(h)$ and the expected risk $R(h)$ for a given sample size m . This yields a regularized inductive criterion:

$$\hat{h}^* = \underset{h \in \mathcal{H}}{\text{ArgMin}} \left[R_m(h) + \text{Capacity}(\mathcal{H}) \right] \quad (8)$$

From this section, one can retain that *the inductive criterion*, be it empirical risk minimization, maximum likelihood or minimum description length principle, *plays an essential role in machine learning*. It both expresses the characteristics of the problem: misclassification costs and measures of performance, and allows us to analyze the conditions for successful induction. In addition, it did motivate several, if not most, modern learners (SVM, boosting, ...). However, it presupposes *a special kind of link between the*

past and the future. The assumption is that both \mathbf{p}_X and $\mathbf{p}_{Y|X}$ are stationary, and that the data are drawn independently and identically from these probability distributions.

While these assumptions were seen as reasonable when most applications involved the analysis of limited static data bases, their validity appear increasingly questionable in face of new learning tasks. This is especially true for knowledge discovery from ubiquitous systems.

3 On-line learning: motivations and issues

Two new trends are shaping the future of machine learning. The first one is that *new types of data sources* are more and more taped in in order to discover regularities or tendencies that help understand the world and make decisions. This is in part due to the growing availability of digitalized observations series (e.g. environmental measurements over time) and to the birth of new sensor systems, that can both be spatially distributed and produce temporal data. Data are thus made increasingly available through unlimited streams that continuously flow, possibly at high speed. Furthermore, the underlying regularities may evolve over time rather than be stationary. Finally, the data is now often spatially as well as time situated. The second trend has to do with the learning systems themselves. They are indeed more and more embedded within complex data processing and management systems that are designed to be “long-life” systems. Consequently, the learning component itself must be able to process the data as it flows in while providing at any given time an hypothesis about the world. This entails that the learning systems must become incremental learning systems.

The overall upshot is that data can therefore no longer be considered i.i.d.. This forces the field to reconsider the basic and fundamental tenets of the theory of induction.

More precisely, the wealth of new applications and learning situations can be categorized into broad classes according to the underlying characteristics of the world.

1. Possibly **stationary world**

- But the learner has *limited resources*. This is the case, for instance, of learning from very large data bases (e.g. Telecoms: millions of examples ; EGEE systems in particle physics: billions of examples, ...)
- But there are “*anytime*” *constraints* on learning that preclude to wait for all the necessary data to be observed and processed before hypotheses or decisions must be made. Data streaming is one typical example of this family of applications.

2. **The target concept is stationary while the distribution \mathbf{p}_X** of the descriptive, also known as explanatory, variables **is changing**. This is called “covariate shift” [12] and is currently the focus of some research effort. Active learning is naturally prone to covariate shift since the learning data, chosen by the learner, is not representative of the underlying distribution.

3. Finally, **the target dependencies $\mathbf{p}_{Y|X}$ themselves might change over time**. This is also known as *concept drift*. This might occur because the world is changing (e.g. the customers’s tastes change with fashion), or because learning is transfered from one task to another one. Another such situation occurs when learning is tutored with the help of a teacher who (carefully) chooses the sequence of learning tasks, with increasingly difficult rules to learn.

4 Aspects of on-line learning

4.1 Reducing computational cost

Since the advent of modern learning algorithms, in the 80s, the common wisdom has been that batch learning was to be preferred to on-line learning. One important reason was that on-line learning tends to be sensitive to the order of presentation of the training examples, which was considered as a nuisance. Furthermore, studies showed that batch gradient algorithms converge much more rapidly to the optimum \hat{h}^* of the empirical risk $R_m(\cdot)$ over a sample of size m , than the corresponding on-line learning algorithms.

More precisely, a *batch gradient algorithm* minimizes the empirical risk $R_m(h)$ using the following formula:

$$h_{k+1} = h_k - \gamma_k \nabla_h R_m(h_k) = h_k - \gamma_k \frac{1}{m} \sum_{i=1}^m \nabla_h \ell(h_k(\mathbf{x}_i), y_i) \quad (9)$$

where the learning rate γ_k is a positive number. Studies have shown that $(h_k - \hat{h}^*)^2$ converges like e^{-k} , where k denotes the k^{th} epoch or sweep over the training set.

By contrast, an *on-line or stochastic gradient procedure* updates the current hypothesis on the basis of a single sample (\mathbf{x}_t, y_t) , usually picked randomly at each iteration.

$$h_{t+1} = h_t - \gamma_t \nabla_h \ell(h_t(\mathbf{x}_t), y_t) \quad (10)$$

Under mild assumptions, on-line algorithms converge almost surely to a local minimum of the empirical risk. But, if they converge to the general area of the optimum at least as fast as batch algorithms, stochastic fluctuations due to the noisy gradient estimate make the hypothesis randomly wobble around the optimum region whose size decreases slowly. The analysis shows that the expectation $\mathbb{E}(h_t - \hat{h}^*)^2$ converges like $1/t$ at best. This seems to condemn on-line algorithms.

However, this study begs two issues. The first one is that, in fact, one is not interested in the convergence to the exact minimum of the empirical risk, but rather in the convergence to the minimum expected risk R . Therefore, fine optimization is not required, and it is the one that is costly for on-line procedures. The second issue is that one should not only compare the convergence with respect to the number of learning steps, but one should also take into account the total computational costs involved. In this respect, on-line learning is quite simple to implement and only involves one random example at each iteration which can be discarded afterwards. On the contrary, each iteration of a batch algorithm involves a large summation over all the available examples, and memory must be allocated to hold these examples and to perform possibly complex computations, for instance if second order derivatives are estimated.

Studies [1, 2] have shown that whereas a batch learning algorithm can process N learning examples, using the same computational resources an on-line learning algorithm can examine T instances, where T is of the order $\mathcal{O}(N \log \log N)$. Thus, for instance, while a batch learner could afford to examine 10,000 instances, a on-line learner could process $\approx 22,200$ instances, or about as twice as much!

This has profound implications for learning from large data sets or from data streams. Indeed, most of the time, learning is mainly limited by the fact that some informative examples have not yet been observed rather than by the fact that the examples already seen have not been fully exploited by the optimization process. When this is true, then **on-line algorithms may turn out to be vastly superior to their batch learning**

counterparts since, for the same computational resources they can process more examples.

This is thus a first reason to seriously consider on-line learning, even when the world is stationary. Of course, this is even more so in face of changing conditions. But first, let us consider some issues raised in incremental learning irrespective of the conditions of the world.

4.2 Incremental learning: issues raised

In principle, it is not difficult to produce an incremental learning system. It suffices to take a batch learner and to make it learn each time this is warranted on the basis of all the training examples seen so far. There are, however, at least two obstacles on the road to the actual implementation of such a scheme. First, this would require a memory size that would grow at least linearly with time (the growth could be worse if the computations required for learning are more than linear on the size of the training set). Additionally, the computation cost can be expected to grow at least in the same proportion. Second, this is ineffective in face of a changing environment because past data may become obsolete and harmful. Control of the memory is therefore required in incremental learning. The question is: how to carry out this control?

As an illustration, suppose we take a very simple incremental supervised learning system, the lazy learner that stores past examples and decides the label of a new unseen instance on the basis of the labels of its k nearest neighbors. After a while, it cannot afford to store all past data, and must per force select instances to be discarded. What should be the optimal forgetting strategy?

There are numerous options, including the following ones:

1. Discard the oldest training instance.
2. Discard the most obvious outlier in the training instances.
3. Discard the instance with the highest proportion of neighbors of the same label.
4. Discard a randomly chosen training instance.
5. Discard a training instance that is the farthest apart from instances of any other class.
6. ...

Each strategy is associated with an implicit model of the world and implies a computational load that may vary between “not worth to mention” (e.g. strategy 4) and “really worrisome” (e.g. strategies 2, 5). Most importantly, there is no best strategy. It all depends on the properties of the varying conditions of the world. Furthermore, as soon as forgetting is allowed to occur, the learning result is prone to be order dependent, that is to depend on the order in which the training instances have been considered [10].

It is worth mentioning that an important part of current research of data streaming systems centers on the question of which *summary* should be kept about past data (see for instance [4]).

To sum up, it was common wisdom that learning required carefully designed search strategies in order to find a (quasi) optimal hypothesis, or in order to chose the most informative examples in active learning. It now appears that **learning should, very generally, require forgetting** as well and that this entails a whole new search and optimization problem in its own right.

Control of the memory will soon emerge as an essential issue, both to limit the space and computational load, but also in order to adapt to the changing conditions of the world. In addition, **sequencing effects will have to be mastered**. It was usual to ignore or to try to reduce them, usually through some randomization process. It may become profitable to use them instead as ways to take into account the history of the environment.

4.3 Covariate shift: changing $\mathbf{p}_{\mathcal{X}}$

Even if the dependencies that link inputs to outputs are stationary, the distribution of the input pattern may vary over time, and, therefore, be different between learning and predicting, a situation called *covariate shift*. For instance, even though the fundamental characteristics of diseases are determined by biological rules and therefore are quite stable, their prevalence may present seasonal variations. Another common situation in learning arises when the distribution of learning instances is tweaked in order to facilitate learning. Thus, one may want to balance the classes of instances when some classes have few representatives, a situation commonly encountered in medical diagnosis. Similarly, in active learning, the training instances are selected by the learner, which, generally, leads to a distorted representation of the true underlying distribution. In these cases, the training data can no longer be considered as independent and distributed according to the true distribution $\mathbf{p}_{\mathcal{X}}$, and, therefore, the measured empirical risk is no longer an empirical measure of the true risk. Of course, all of this misrepresentation of the training data is ever more true with data streams corresponding to an evolving phenomenon.

In this case, it is known that the classical inductive criteria, such as (regularized) empirical risk minimization or the maximum likelihood estimator, lose their consistency: the learnt estimator or hypothesis does not converge in probability to the optimal one.

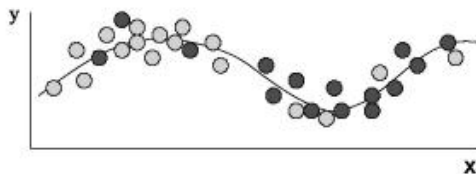


Figure 2: In a stationary environment where the dependency $\mathbf{p}(y|\mathbf{x})$ is stationary, it may happen that the learning and the testing samples (respectively in light gray and dark gray) are not drawn from the same distribution (borrowed from [9]).

Suppose $\mathbf{p}_{\mathcal{X}}$ denotes the training distribution and $\mathbf{p}_{\mathcal{X}'}$ the test distribution. The performance of the hypothesis learnt using a training sample drawn according to $\mathbf{p}_{\mathcal{X}}$ depends on:

- the performance of the hypothesis over $\mathbf{p}_{\mathcal{X}}$
- the similarity between $\mathbf{p}_{\mathcal{X}}$ and $\mathbf{p}_{\mathcal{X}'}$.

One obvious solution to regain consistency is to weight the training instances according to what is called their *importance*, that is the ratio of the test and training input densities: $\mathbf{p}_{\mathcal{X}'}(\mathbf{x})/\mathbf{p}_{\mathcal{X}}(\mathbf{x})$. One then gets the *importance weighted ERM* [12]:

$$\hat{h}^* = \underset{h \in \mathcal{H}}{\text{ArgMin}} \left[\frac{1}{m} \sum_{i=1}^m \frac{\mathbf{p}_{\mathcal{X}'}(\mathbf{x}_i)}{\mathbf{p}_{\mathcal{X}}(\mathbf{x}_i)} \ell(h(\mathbf{x}_i), y_i) \right] \quad (11)$$

Apart from stability considerations that imposes some modification, this new inductive criterion necessitates the estimation of the importance $\mathbf{p}_{\mathcal{X}'}(\mathbf{x})/\mathbf{p}_{\mathcal{X}}(\mathbf{x})$. However, the naive approach which is to first estimate the training and test input densities and then compute their ratio is rather impractical since density estimation is notoriously hard, especially in high dimensional cases. This is why recent research efforts have aimed at directly estimating the importance, bypassing density estimations. See e.g. [13] for more details.

4.4 Concept drift: changing $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}$

Because of the increasing availability of data streams gathered over a long stretches of time, handling changing distributions and concept drifts has become a new important topic in machine learning. For instance, a company may find that her customer profile is varying over time. Likewise, in document filtering applications, the interests of the users may drift, or even abruptly change. The learning system should then revise and continuously adapt its model accordingly. We restrict ourselves here to the pattern recognition framework, that is to supervised learning of classes of patterns.

The learning problem may be characterized as follows. We suppose that data arrives in sequence, either one at a time or in small batches. Within each batch the data is independently and identically distributed with respect to a “local” distribution $\mathbf{p}_{\mathcal{X}\mathcal{Y}}(t)$. A concept drift occurs when the conditional distribution $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}$ changes with time. The aim of the learner is to sequentially predict the label of the next example or batch, and to minimize the cumulated loss. Often this cumulated loss will correspond to the number of prediction errors.

It is assumed that the newly arrived data most closely resemble the current true concept. Furthermore, a reasonable assumption is that there exists some sort of temporal consistency in the changing environment corresponding to the fact that, most of the time, the underlying distribution of the data is changing continuously. These two assumptions imply that recent data carries more information about the current underlying concept than more ancient data.

Accordingly, learning in a changing environment is often handled by *keeping windows* of fixed or adaptive length on the data stream, or by *weighting data* or parts of the current hypothesis in accordance with their age or their utility for the learning task.

Whatever the approach, maintaining sliding windows or weights, the same tradeoff must be solved. On one hand, the system must be able to detect true variations against a noisy background, meaning it must be robust to irrelevant variations. On the other hand, the system should adapt as quickly as possible to variations in the environment in order to minimize its cumulated loss. Unfortunately, these two demands point to opposite strategies. Robustness to noise increases with the amount of data that is taken into account, but, in changing conditions, the oldest the data, the more likely it is obsolete. Therefore, at the same time, one would want to keep as much as possible information from the past, while reducing its importance for fear of being erroneously biased. Most works in concept drift have focused on devising heuristics to solve this conundrum, that is **to control the memory of the past**.

Domingos and Hulten, [5], attack head on the issue of learning in face of very rapid data streams. They require their Very Fast Decision Tree learner to induce a decision tree on the flight so that the result is with high probability almost the same as the one that would be obtained with a batch learner but using only constant time and space complexity. For this, they rely on the statistical theory of learning, specifically on Hoeffding formula, in order to compute bounds on the required minimal training set

size to approximate the optimal decision function within a given error factor. Here, this bound is iteratively used for the determination of each node in the tree. Of course, this learning methods uses much more data than would be strictly necessary to ensure to get a good approximation of the target tree, but it is assumed that there is an over-supply of data. The interesting idea in this approach therefore lies in using the theory to get estimates of the required window size on the past sequence of data. The limit, however, is that it assumes stationarity. In a subsequent paper, [7], the problem of time-changing environment was tackled and another useful concept was put forward: to grow a new tree when the data is starting to drift away from the previous distribution and to start using the new tree when it becomes more accurate than the old one. This enables to use past information as long as it is useful, and to overcome to some degree an explicit trade-off for the choice of the window size.

Managing window size

If a fixed size is to be chosen for sliding windows, the choice results from a compromise between fast adaptability (small window to the risk of under-fitting) and good generalization (large window). It can only be made on the basis of assumptions about the pace of the changes in the environment. If no such well-informed assumption is possible, one has to rely on adaptive strategies for the window size management. The challenge in automatically adjusting the size of the window is to minimize at each time the expected loss on new examples. This requires that the model of the data is as accurate as possible at each time step. When the underlying distribution is stable, the window size can safely increase, enabling better generalization and the learning of more detailed and accurate models of the environment, whereas when the distribution is changing, the window size must shorten in order to not incorporate obsolete and harmful training instances (see figure 3).

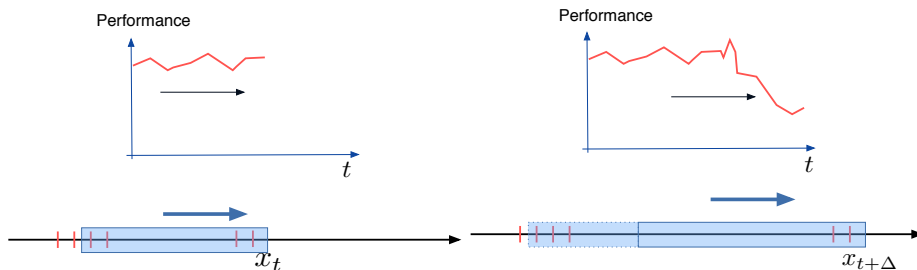


Figure 3: When the underlying distribution is stable, or at least, the performance is, the size of the window may increase (left). When the performance is falling, betraying a change in the distribution, the window size must be reduced (right).

One way to implement an adaptive strategy goes as follows. Suppose that batches of equal size arrive at each time step. Suppose also that the current time step is T while the learning process started at time $t = 0$. A classifier is learnt over the most recent batch, and is tested over every preceding windows of size less than $\max(\text{max_length}, T)$. The window of maximal size for which the error is $< \varepsilon$ for a given ε is kept. A classifier is learned over this window and is used for predicting the class of the newly arriving unclassified example(s) (see figure 3). In case of a abrupt change of distribution, it may happen that the learning window is reduced to the most recent batch of data.

One problem with this strategy is that there is no memory of the past when the

underlying distribution has changed. Let us suppose, for instance, that the underlying distribution switches from distribution $\mathbf{p}_{y|\mathcal{X}}^1$ to distribution $\mathbf{p}_{y|\mathcal{X}}^2$, and then reverses back to distribution $\mathbf{p}_{y|\mathcal{X}}^1$. Then, when the first distribution $\mathbf{p}_{y|\mathcal{X}}^1$ rules the generation of data once more, learning will have to restart all over again. This is why other approaches have been proposed [11]. For instance, one may envision that all past batches for which the prediction error of the classifier learnt over the most recent batch is $< \varepsilon$ are kept for learning the classifier used for prediction (see figure 4). In this way, past data that seems relevant to the current learning situation may be used for learning. The relearning time may thus be greatly reduced, enabling better performance.

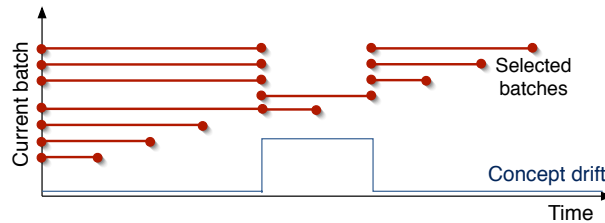


Figure 4: Adaptive management of window size with memory if a concept is encountered again. (Borrowed from [11]).

Weighting past examples

There have been very numerous proposals for selecting or weighting past instances in order to confront changing environments. For the purpose of illustration, we mention here one technique called “locally weighted forgetting” and used in nearest neighbors classification. In this scheme, all data starts with weight 1, and when a new data point is observed, the weights of the k nearest neighbors are adjusted according to the following rule: (1) The closer the data to the new sample, the more the weight is decayed; (2) If weight drops below some threshold, remove data.

In this way, it is hoped that only sufficiently scattered representatives of the data are kept.

But it is also possible to maintain weights on hypotheses rather than on data points.

Weighting past hypotheses

In the wake of the success of boosting techniques, ensemble methods for tracking concept drift have been recently proposed. The overall strategy is the following.

1. Learn a number of models on different parts of the data.
2. Weigh classifiers according to recent performance.
3. If classifier performance degrades, replace it by a new classifier.

More specifically, for instance, Kolter and Maloof in [8] describe the “dynamic weighted majority” algorithm that dynamically creates and removes weighted classifiers in response to changes in performance.

1. Classifiers in ensemble have initially a weight of 1
2. For each new instance:

- If a classifier predicts incorrectly, reduce its weight
- If weight drops below threshold, remove classifier
- If the ensemble of current classifiers then predicts incorrectly, install new classifier
- Finally, all classifiers are (incrementally) updated by considering new instance

Among other proposals involving ensemble methods, like [15], one is standing out by introducing an intriguing and tempting idea [11]. Suppose that the underlying data distribution is continuously changing from concept 1 to concept 2, is it possible to learn concept 2 before the end of the transition? (See figure 5). Under some admittedly restrictive assumptions, this turns out indeed to be possible.

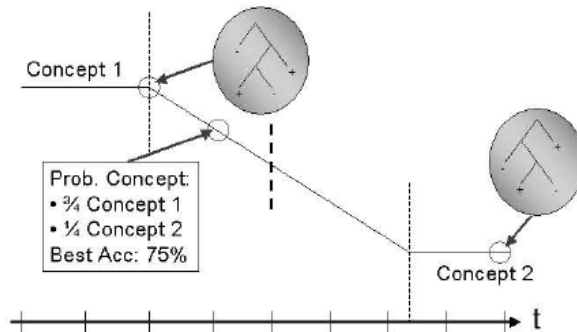


Figure 5: Continuous concept drift changing linearly from concept 1 to concept 2. It is optimal to predict concept 1 before the dotted line, and concept 2 afterwards. (Borrowed from [11]).

The main assumption is that, during the concept drift, the training instances are sampled from a mixture distribution, that is a weighted combination of the two pure distributions characterizing concept 1 and concept 2. The optimal model during the concept drift can only be derived according to the Bayes's optimal rule. The idea is to decompose the mixture distribution as soon as the concept drift starts. In this respect, the boosting algorithm is very seducing since it is based on the principle to sample the data points at each step *orthogonally* to the current distribution. In the approach of Scholz and Klinkenberg, this becomes sampling orthogonally to the distribution induced by the prediction of h_1 , the hypothesis learnt from data corresponding to concept 1. A careful analysis leads to a weighting scheme that modifies the weight of the examples with respect to hypothesis h_1 so that the new distribution reflects the characteristics of the new incoming distribution.

Lessons about concept drift

Learning in the presence of concept drift is still very much an open research issue even though a lot of interesting ideas and heuristics have been put forward. Overall, the existing methods are sometimes efficient in their respective application domains, but they usually require fine tuning. Furthermore, they are still not easily transferable to other domains. This denotes a lack of a satisfying theoretical ground.

There are indeed relatively few theoretical analyses, and most of them date back to the early 90s. One significant work is the one by Helmbold and Long [6]. They

study the conditions for PAC learning with an error of ε in the presence of concept drift. This depends upon the diversity of the hypothesis space \mathcal{H} and on the speed of the drift (measured as the probability that the 2 subsequent concepts disagree on a randomly drawn example). The outcome is a bound of the size of the required window size. Regrettably, these bounds are usually impractically large. This is due in part to the adversary protocol used in the analysis.

Besides the shortcoming of our current theoretical understanding, there are other desirable developments. Among them is the need for the capability to recognize and treat recurring contexts, for instance associated with seasonal variations, so that old models can be quickly recovered if appropriate. But, more significantly, there is a growing feeling that it would be profitable to **focus on the changes themselves** rather than merely trying to follow concept drifts as closely as possible. Reasoning about the “second derivatives” of the evolving situation and representing them would allow for quicker adaptation, as well as interpretability about what has changed and how. This raises the issue of having models for change and to incorporate them in new appropriate inductive criteria.

5 A new perspective on on-line learning

In the following, we focus on a special case of learning task that has been conjured up in a recent paper by Sutton, Koop and Silver [14], called *tracking*.

5.1 The tracking problem

Sutton et al. argue that while most existing learning systems are geared to find a single best solution to the learning problem, one that applies to any possible input $\mathbf{x} \in \mathcal{X}$, it might be possible that better performance be attained with the same amount of training data and computing resources by tracking the current situation rather than by searching the best overall model of the world. In this view, the agent continuously learns a local model that applies to the situations that can be encountered at the time being.

More precisely, it is assumed that the learning agent encounters different parts of the environment at different times. The underlying distribution on \mathcal{X} is now a function of time: $\mathbf{p}_{\mathcal{X}}(t)$, in such a manner that the data are not identically and independently distributed, but are governed by some time dependent process, like, for instance, a Markov decision process. In this case, it might be advantageous for the agent to adapt to the local environment defined by $\mathbf{p}_{\mathcal{X}}(t)$ and, possibly, by $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}(t)$, if the later one is evolving too. Figure 6 schematizes the evolution of such a data-driven agent.

Temporal consistency, which we loosely define as the fact that $\mathbf{p}_{\mathcal{X}}(t)$ and $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}(t)$ tend to evolve with cumulated bounded variation over limited periods of time, offers the opportunity to perform well in term of predictions by learning simple models with limited resources. Indeed, because of temporal consistency, the learner may expect that it will have to make predictions about inputs that lies in its “local environment”. In addition, temporal consistency imposes that the laws governing the local environment are simpler than the laws governing the whole input space and the whole time evolution of the world. Thus, even though the overall model of the world and it’s time evolution may be arbitrary complex, it can be expected that, locally, in term of both input space and time, simple models may suffice for appropriate decisions.

Figure 7 illustrates this in a simple but extreme case. Suppose that $\mathcal{X} = \mathbb{R}$, and that the target dependency $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}(t)$ is stationary and takes the form of a piecewise linear

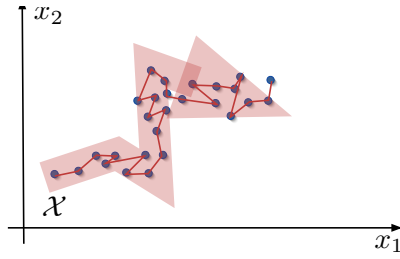


Figure 6: In tracking, the learning agent receives input that are driven by a time dependent process. It therefore encounters different parts of the environment at different times.

curve. Suppose that the local environment of the learning agent may be depicted by a window of limited size. As the agent is exploring \mathcal{X} , passively or actively, it may perform rather accurate predictions solely by maintaining a model of the world that is simply a “constant” prediction. This constant is updated with time, a process described by *tracking*.

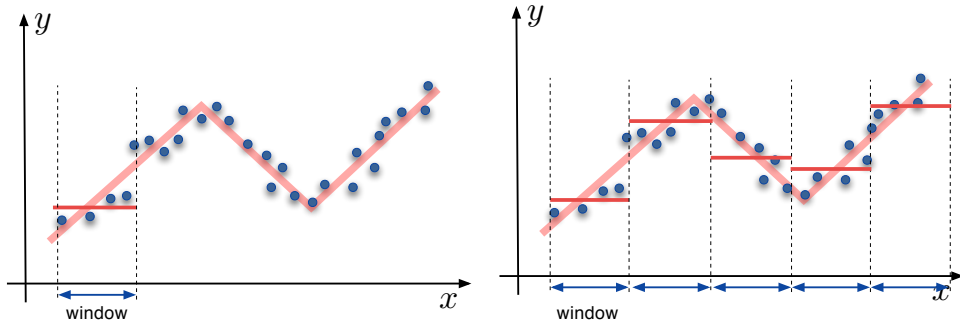
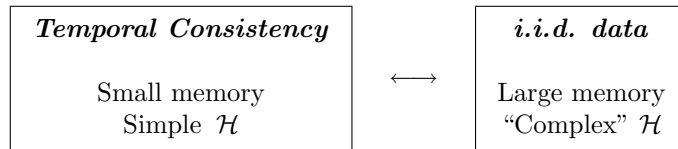


Figure 7: Even though the world involves a piecewise linear law, the learning agent may perform well by maintaining a very simple model, a constant, over its local environment.

Therefore, if time consistency holds, good prediction performance can be obtained with less computational cost than the classical batch learning. At each time step, the learner is searching for simpler models of the world and does so on the basis of limited amount of past data. There is thus a kind of spectrum to be expected along the following lines:



If, intuitively, a tracking strategy seems advantageous, several questions remain to be answered. The tracking problem, well-known in engineering sciences, but new in machine learning, needs to be formally defined. More importantly, we do not know yet how to measure the position of a learning problem along the afore-mentioned spectrum and how to evaluate the advantage in terms of learning resources needed for a given

performance level in terms of prediction. In fact, the classical notion of expected performance and the associated risk formula certainly needs to be revisited. Finally, all of this, yet to come, analysis should be turned into new learning strategies.

In the following, we just propose a glimpse of the kind of new inductive criteria that could be interesting to consider.

5.2 A new perspective for on-line induction

In the classical stationary framework, where learning is supposed to occur once for all from the available training data, the expected performance is defined with respect to each possible model of the world h :

$$R(h) = \mathbb{E}[\ell(h(\mathbf{x}), y)] = \int_{\mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}} \ell(h(\mathbf{x}), y) \mathbf{p}_{\mathcal{X}\mathcal{Y}} d(\mathbf{x}, y)$$

which gives rise to the inductive criterion based on empirical risk:

$$R_m(h) = \frac{1}{m} \sum_{i=1}^m \ell(h(\mathbf{x}_i), y_i)$$

With on-line learning of successive hypotheses, these criteria can no longer be used as such. While it is clear that the performance still involves a cumulated loss over time, neither the model of the world h , nor the underlying data distribution $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$ are stationary. It is therefore necessary to include their variations in the performance and inductive criteria. In fact, not only to include these variations, but to make them the focus of the optimization problem.

Indeed, the very notion of training sample needs to be cross-examined. In on-line learning, it is rarely the case that the learning system will be submitted twice to the same kind of *history*. Therefore, past data cannot be considered as representative of what will happen next. In other words, all past distributions $\mathbf{p}_{\mathcal{X}}(t-k)$, and, possibly as well, all past $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}(t-k)$ for $0 \leq k \leq t$ may very well never be encountered again.

But, then, what would link the past to the future and allow for prediction and induction? One inductive assumption is that the underlying regularity to be learned is the rule that governs the variations of the environment. Denote r the model of the rule that the learner tries to estimate, then the risk associated with such a rule can be expressed as:

$$R(r) = \mathbb{E}[\ell(h_t(\mathbf{x}_t), y_t)] = \int_t \ell(h_t(\mathbf{x}_t), y_t) \mathbf{p}_{\mathcal{X}\mathcal{Y}}(t) d(\mathbf{x}_t, y_t) \quad (12)$$

While the rule r does not appear explicitly in the above expression, it is nonetheless present by the fact that the successive hypotheses h_t are linked by the rule:

$$h_{t+dt} = r(h_t, (\mathbf{x}_t, y_t), \text{memory}, \ell, dt) \quad (13)$$

or, if time flows in discrete time steps:

$$h_{t+1} = r(h_t, (\mathbf{x}_t, y_t), \text{memory}, \ell) \quad (14)$$

Here, the *memory* term is used to denote what trace of the past data is used by the update rule r .

Then, a possible inductive criterion, based on past data from time 0 to time T , could be of the form:

$$L_{(0,T)}(r) = \underbrace{\sum_{t=1}^T \ell(h_t(\mathbf{x}_t), y_t)}_{\text{classical cumulated loss}} + \lambda \underbrace{\sum_{t=1}^T \|h_t - h_{t-1}\|^2 + \text{Capacity}(\mathcal{R})}_{\text{new criterion on } r} \quad (15)$$

Where $\lambda > 0$ is a parameter weighting the importance of conditions over the regularity of the rule r . This regularity is conditioned both by the cumulated variations over h_t (temporal consistency imposes limited variations), and by the complexity of the possible rules. Indeed, the *capacity* is a function of the *memory* used for updating the current hypothesis at each time step. This memory should be kept as limited as possible.

We do not delve into details within the limited scope of this chapter, but it is obvious that this kind of criterion is related to the theory of reinforcement learning and the underlying assumption of Markov Decision Process. There also, what the learning agent is estimating are the rules that govern the transition from one state to the following and the reward function attached to states transitions³.

6 Conclusions

Recent years have witnessed a wealth of emerging applications that can not be solved within the classical inductive setting. New learning tasks often involve data coming in unlimited streams and long-life learning systems that, in addition, have limited computational resources. The fact that data can no longer be considered as identically and independently distributed, and that the learner needs, per force, to implement on-line learning raises important new issues and announces profound evolutions of the field of machine learning.

Among the list of open questions are the following ones:

- How to deal with non i.i.d. data?
 - What to memorize? / What to forget?
 - How to cope with or take advantage of *ordering effects*?
 - How to *facilitate future learning*, what should be transferred? Representations, learned rules, ...?
- What should the inductive criterion be?
 - How to take the *computational resources* into the inductive criterion?
 - What kind of regularity should we optimize: $h \in \mathcal{H}$ or $r \in \mathcal{R}$?

There is already a growing body of work that touches on these questions. *Covariate shift, transduction, concept drift, tracking, transfer of learning, even teachability*, are subject matters that bear on the issue of on-line learning.

One important clue seems to be that, in evolving environments, the changes themselves should be the focus of learning. Works in concept drift have shown that this can accelerate recovery of useful past regularity, but, more generally, the analysis of possible

³ It must be noticed that the problem of the performance criterion is approached quite differently in the theory of on-line learning based on regret criteria. For lack of space, we defer the reader to [3].

new inductive criteria adapted to the problem of on-line learning seem to point to that direction as well.

In any case, whatever will be its scientific outcome, the present time is a privileged one for machine learning, a time for exciting research both for a better fundamental understanding of learning and for the design of new learning techniques. Ubiquitous learning environments, specially, are both the fuel and the beneficiaries of these incoming developments.

References

- [1] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (<http://books.nips.cc>), 2008.
- [2] Léon Bottou and Yann LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.
- [3] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning and games*. Cambridge University Press, 2006.
- [4] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.
- [5] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM Press, 2000.
- [6] D. Helmbold and P. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1):27–45, 1994.
- [7] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, New York, NY, USA, 2001. ACM.
- [8] J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, 2007.
- [9] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence. *Dataset shift in machine learning*. MIT Press, 2009.
- [10] F. Ritter, J. Nerb, E. Lehtinen, and T. O’Shea, editors. *In order to learn. How the sequence of topics influences learning*. Oxford University Press, 2007.
- [11] Martin Scholz and Ralf Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007.
- [12] M. Sugiyama, M. Kraudelat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, 2007.

- [13] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.
- [14] R. Sutton, A. Koop, and D. Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, pages 871–878, Corvallis, Oregon, 2007. ACM.
- [15] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining (KDD'2003)*, pages 226–235, 2003.