WOAS WORLD ORGANIZATION OF APPLIED SCIENCES JOURNALS

International Journal of Information Technology and Applied Sciences

A simple formalism Artificial intelligence-based to represent knowledge in a multi-agent planning context

Imane Ben

Higher Institute of Computer Science and Multimedia of Sfax, Tunisie

Abstract: At the start of the simulation, the agent knows nothing about how the dynamics of interaction with the environment unfold, or what causes his sensations. He does not distinguish obstacles from free paths, and he does not know the consequences implied by his actions. Under these conditions, the CALM mechanism was able to converge steadily towards the expected solution, by building a model of the world adequate to represent the regularities of the environment, the regularities of its bodily sensations, as well as to represent the influence regular actions on both. The agent learns about the consequences of his actions in different situations, which are represented by a reduced number of very general diagrams. From them, the mechanism can build an action policy that allows it to avoid affectively negative situations and to seek those that are affectively positive. This solution manages to describe precisely all the regularities that the agent can perceive without building a complete plan of the environment.

Keywords: Artificial intelligence, CALM., machine learning.

1. INTRODUCTION

Intrusion detection systems are traditionally based on signatures generated manually by security experts. However, Big Data, artificial intelligence, Machine Learning (automatic learning in French) or Deep Learning (deep learning in French) are often presented as technologies that can revolutionize intrusion detection systems [1,5]. Indeed, these methods induce detection rules automatically from data, and their generalization capabilities allow them to detect malicious events that are still unknown. Many research papers on the application of machine learning to intrusion detection have been published and often show exceptional results (detection of malicious PDFs [6], of malicious executable files [4], or botnets [2,3]). However, from an operational point of view, there are still many reservations about the use of machine learning in production:

- A detection system based on learning methods automatic can it process in real time?
- Is the false positive rate of machine learning methods, often presumed to be too high, acceptable for them to be put into production?
- Machine learning is not the core business of a security expert, yet it is on him that the implementation of a detection system rests. How can he trust these methods to put them into production?
- Are the alerts generated by such a detection system sufficiently interpretable to allow their exploitation in production?

In this paper, we answer the questions mentioned above by providing solutions so that machine learning methods are not incompatible with operational constraints, and that they can thus be integrated into detection systems in addition to other methods. , like the signatures. First, we briefly present the problem of intrusion detection and give a general overview of machine learning in this context (see section 2). Then, we present a case study: the detection of malicious PDF files (see section 3). This example will then be used to illustrate good practices for learning and validating a detection model, and to highlight pitfalls sometimes omitted in academic publications (see sections 4 and 5). Finally, we present the free SecuML tool (cf. section 6) allowing in particular to build a detection model and to validate it upstream of its production with the good practices exposed in the paper.

2. INTRUSION DETECTION AND MACHINE LEARNING

The role of an intrusion detection system is to detect malicious events through the network and system activities that it analyzes. A suspicious event can be the recovery of a malicious file attached to an email or the visit to a corrupt website for example.

E-mail:author's email

The detection system administrator is notably responsible for setting up detection methods, and for developing them over time. The security operator analyzes and qualifies the alerts in order to allow the necessary measures to be taken to deal with any possible security incidents. This work can be costly.

Intrusion detection systems are traditionally based on signatures: detection rules constructed by an expert following an in-depth analysis of malicious events. This approach is effective against threats that have already been observed and for which a signature has been generated, but it is often ineffective in detecting new threats. In addition, simple variations of the threat, such as polymorphism [5], may be enough to render the signature ineffective. Signatures are ubiquitous in current detection systems, but detection methods with machine learning are considered in addition to better detect new threats. In this section, we present the two main categories of machine learning methods that can complement the signature approach: anomaly detection and supervised learning.

A. Anomaly detection

Anomaly detection is the first machine learning method that has been applied to intrusion detection [2]. This approach only requires non-malicious data to build the detection model. The model will then generate an alert as soon as an event differs too much from the normal behavior induced by the benign data initially provided.

Anomaly detection methods are very attractive because they allow the detection of unknown threats. They have no preconceptions about what is a malicious event and are therefore prone to detect new threats. In addition, putting them into production is often presented as very simple: all you need is a benign dataset devoid of malicious activity. Obtaining such a dataset is not easy in practice, however, as there is no simple way to ensure that there is no malicious activity. If the supposedly healthy dataset contains malicious activity, this can distort the learning of the model and prevent the detection of certain threats. These detection systems are simple in principle, but rarely in practice, and putting them into production can be very complex. In addition, anomaly detection methods raise alerts for abnormal events that are not necessarily malicious. For example, an abnormal transmission / reception ratio on HTTPS can be a sign of data exfiltration, but can also be caused by the use of certain social networks; popular websites can be the source of seemingly abnormally large data traffic without being malicious; and simple configuration errors can also lead to behaviors triggering false alerts. Thus, these detection methods often suffer from a high rate of false positives. Finally, the detection of anomalies offers few possibilities for taking into account expert knowledge. Indeed, experts cannot guide these models by providing examples of malicious events as they only take into account mild events.

B. Supervised learning

Supervised learning responds to this need to integrate expert knowledge. Indeed, a supervised detection model is constructed from labeled data provided by the expert: mild events, but also malicious events to guide the detection model. The learning algorithm will automatically look for the points allowing to characterize each of the classes or to discriminate them to build the detection model. Once the detection model is learned on a training dataset, it can be applied automatically to detect malicious events. Thanks to supervised learning, the security operator supervising the detection system can easily participate in improving the detection model based on the alerts that he analyzes. Indeed, false alerts can be reinjected to correct the detection model and thus avoid generating the same false alerts in the future. The real alerts can also be fed back into the model to let it follow the evolution of the threat. Thus, security experts do not give control of the detection system to an automatic model, but they actively supervise it to improve its performance over time [5].

Finally, supervised learning is guided by malicious examples provided by the expert, which reduces the rate of false positives compared to the detection of anomalies. Supervised methods are therefore to be preferred when labeled data are available to train the detection model. However, these methods must be applied taking into account the operational constraints of the detection systems. The detection model must be able to process data in real time, and the false positive rate must remain below a certain threshold to prevent the security operator from being overwhelmed by false alerts. Finally, the administrator must have confidence in the model to put it into production, and the operator must be able to understand the alerts generated. In the rest of the paper, we give a methodology so that machine learning meets these constraints, and so that it can be integrated into detection systems to better detect new threats.

3. CASE STUDY: DETECTING MALICIOUS PDF FILES

In this section, we present the problem of detecting malicious PDF files based on machine learning. The rest of the article will use this case study to illustrate good practices and highlight the pitfalls to avoid when using machine learning to build a detection model.

A. Problem

The PDF format is an open document description format, created by the Adobe company in 1993, aimed at preserving the formatting regardless of the reading software or the operating system used. Among other things, it consists of a number of metadata such as the author, the date of production, as well as objects of different types referenced in a table called Xref. These objects can be in particular text, images, video or even JavaScript code. Its richness and the availability of readers

on different platforms make it a format widely used in most organizations for creating and exchanging electronic documents. On the other hand, the volume of associated specifications (more than 1,300 pages available publicly) implies significant software complexity, amplified by dependencies with numerous third-party libraries. Also, this software is often subject to vulnerabilities, which make PDF format even more attractive to attackers.

In the majority of cases, malicious PDF files are forged by the attacker to exploit a vulnerability, in order to execute code and compromise the victim's machine (for example JavaScript code exploiting a vulnerability of the JavaScript engine included in the reader, or a TTF font exploiting an OS vulnerability). Among the elements that we can try to locate to detect malicious PDF files, we can note:

the presence of a malicious charge (a shellcode, etc.);

- Functions to deceive detection (obfuscation by encryption, multiple encodings, concealment of objects, etc.).
- The more or less realistic nature of the files (malformations, low number of pages and / or objects, etc.).

Supervised learning is preferred in the context of intrusion detection systems (see section 2), and it is easy to obtain benign and malicious PDF files using Contagio3, VirusTotal4, or the Google search for example. So we chose supervised learning to build a model for detecting malicious PDF files. The following section outlines the main steps in the use of supervised learning, and describes the performance estimators for properly evaluating a detection model.

B. Supervised learning

Two phases: learning and prediction Supervised learning can be used for intrusion detection via a binary classifier (see Figure 1). The classifier takes as input an instance, a PDF file for example, and returns as output the predicted label, benign or malicious (in green and red in the figure). This classifier can also be called a detection model in the context of intrusion detection.

Supervised learning has two main stages:

1. learning the classifier from labeled data.

2. use of the classifier to detect malicious instances.

4. APPROACH AND METHODOLOGY

Once the classifier is trained from the training data, it can be used to predict the label of a PDF file. In practice, most classifiers do not simply predict a binary value (benign vs. malicious), but rather a probability of maliciousness (see Figure 3). An alert is then generated only if the probability of malicious attack is greater than the detection threshold set by the administrator of the detection system. In the example in Figure 3, an alert will be raised for the PDF file considered only if the detection threshold is less than 75%. The probability of malicious acts predicted by the detection model makes it possible to classify the alerts according to the trust of the model, and therefore to define the priority of the alerts that the operator supervising the detection system must deal with. Performance estimators A detection model is not perfect; it can make prediction errors. It is essential to validate it, that is to say to measure the relevance of the alerts generated, before putting it into production.

The best known performance estimator is the classification error rate which is equal to the percentage of poorly classified instances. However, in the case of intrusion detection, the data is generally very asymmetrical (with a small proportion of malicious instances), and the error rate is not able to correctly estimate the performance of a classifier. in this situation. Here is an example showing the limits of the classification error rate. We consider 100 instances: 2 malicious and 98 benign. In this situation, a foolish detection model that always predicts benign will have a classification error rate of only 2% when it is not able to detect the slightest malicious instance. In order to correctly analyze the performance of a detection model, the first step consists in writing the confusion matrix which takes into account the two types of possible errors: false positives, i.e. false alerts lifted for benign instances, and false negatives, that is, undetected malicious instances. Figure 2 explains the content of a confusion matrix.

A detection model must be evaluated with these two performance estimators taken together. Indeed, the rate of false positives must be low so that the security operator supervising the detection system is not overwhelmed by false alerts. The detection rate must be high to avoid too many threats remaining undetected. The detection threshold determines the sensitivity of the detection: lowering this threshold increases the detection rate, but also the rate of false alerts. It is therefore set by the detection system administrator according to the desired compromise between detection rate and false positive rate. The performance estimators that we have just presented depend on the value of the detection threshold. Another performance estimator, which has the advantage of being independent of this threshold, is often used in detection: the receiver efficiency function, more frequently referred to as the ROC5 curve [9]. This curve represents the detection rate as a function of the false positive rate for various detection thresholds (see Figure 5). For a threshold of 100%, the detection and false alarm rates are zero, and for a threshold of 0% they are both at 100%. A detection model is all the more efficient as its curve is close to the upper left corner: a high detection rate for a low rate of false alerts. The area under the ROC curve, called AUC6, is often calculated to estimate the performance of a detection model independently of the detection threshold, and its value must be close to 1. A classifier randomly predicting the probability of malicious attack a for ROC curve the red line represented in figure

3. Thus, the ROC curve of a classifier must always be above this line (otherwise a random classifier has better performance ...), and the AUC is at a minimum 0.5. The ROC curve is not only a performance estimator, but it also allows the detection system administrator to choose the detection threshold value according to the desired detection rate or the tolerated false alarm rate. Generic method Our presentation of supervised learning was based on the example of the detection of malicious PDF files, but the instance can also represent a DOC file, traffic associated with an IP address or a web page for example. Machine learning algorithms do not take raw instances as input, but a representation in the form of vectors of digital attributes7 of fixed size. Thanks to this representation of instances, machine learning algorithms are generic and can be easily applied to various intrusion detection problems. The attribute extraction step is specific to each detection problem. In the following section, we present the main steps that must be taken to create a supervised detection model by illustrating our remarks with the case of the detection of malicious PDF files.

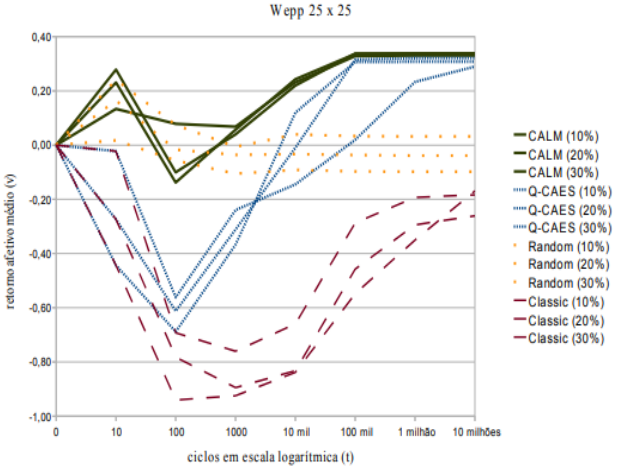

Figure 1. Wepp 125 x 125: Q-Learning x CALM 1.

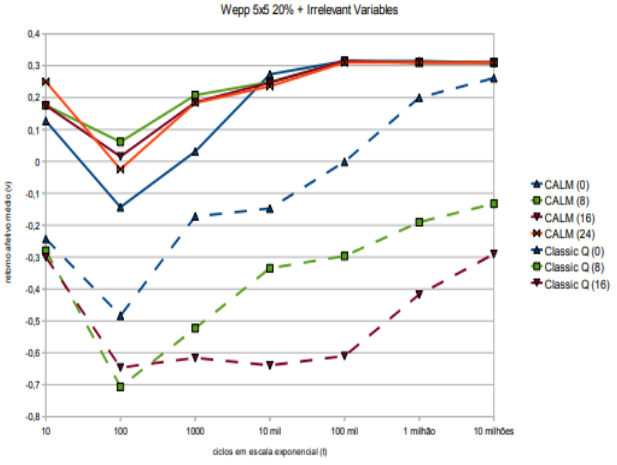

Figure 2. Extensibility analysis.

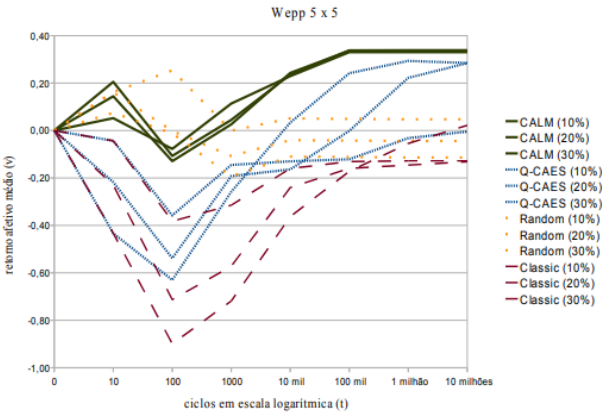

Figure 3. Wepp 125 x 125: Q-Learning x CALM 2.

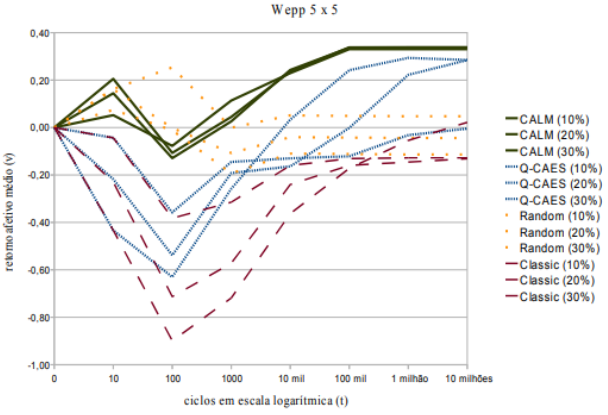

Figure 4. Wepp 125 x 125: Q-Learning x CALM 3.

5. How to build a detection model?

The first step before building a detection model with machine learning is to define the target, i.e. what you want to detect. This preliminary step was done for the problem of detecting malicious PDF files in section 3.A. Then, to create a detection model, you must:

- collect learning data containing benign and malicious instances corresponding to the target.
- define the attributes to extract to represent the instances in the form of digital vectors.

Choose a type of classification model adapted to operational constraints. We now describe these three steps with generic tips and examples from the PDF file case study.

A. Obtain learning data

Building a supervised detection model requires learning data for which the labels, benign or malicious, are known. This training data must contain benign and malicious instances corresponding to the detection target. However, obtaining learning data is often expensive,

because associating a label with instances requires the knowledge of a security expert. There are publicly labeled data sets for certain detection problems (Malicia project, KDD99, kyoto2006, or Contagio for example), but for other problems, a data annotation phase must precede the construction of the model. This step can be tedious, but the effectiveness of the model will be directly linked to the composition of the learning game, this is why this step should not be overlooked. There are a few main principles to follow when building the learning game. First, it must have a sufficient number of instances for the detection model to be able to properly generalize benign and malicious behavior. Typically, it seems very difficult to learn a model if the training data contains less than a few hundred instances for each label. Furthermore, care must be taken not to have an overly unbalanced training data set with a label representing a very small proportion of the data. It is impossible to define general rules concerning the minimum number of instances necessary to learn a model, or the acceptable degree of imbalance, because these values depend on the detection problem considered and on the instances constituting the learning game.

In the case of detection of malicious PDF files, it is easy to obtain a labeled data set, since this type of file is popular and frequently used to propagate malicious code. The experiences presented in this paper are based on two data sets: Contagio (9,000 benign files and 11,101 malicious) and webPdf (2,078 benign files and 767 malicious). Contagio is a public dataset used in many academic works, and we built webPdf from benign files from the Google search engine and from malicious files obtained on the VirusTotal platform.

B. Extracting discriminating attributes

This step consists in processing the data in order to represent it in the form of vectors of digital attributes of fixed size usable by the learning algorithms. Attributes are digital characteristics extracted from the data that will allow the decision making of the classifier: the more they are discriminating for decision making, the more efficient the classifier will be. It is therefore necessary to have a good knowledge of the format and content of the data considered, and to have well defined the detection target to extract discriminating attributes. The attribute extraction phase is specific to each detection problem, but common techniques for extracting digital attributes can often be applied. We present some classic methods of extracting attributes, giving examples with the case of PDF files.

PDF files contain two types of information that can be used to generate attributes: metadata (author or date of creation, for example), and a list of its objects. Some information is already digital, or a simple transformation can make it digital. For example, the file size is numeric and the creation and modification dates can be transformed into timestamps. However, other information, such as the author or objects, is not numeric, and therefore cannot be directly exploited by machine learning algorithms. In addition, each PDF file has a variable number of objects: how to represent this information in the form of a vector of fixed size?

Character strings Discriminating information can take the form of character strings. The extraction method that we used consists in transforming a character string into a vector of attributes where each attribute corresponds to a family of characters (capital letters, small letters, or numbers for example). The value of an attribute is determined by the number of occurrences, or the proportion, of this family in the character string. The author of a PDF file is a character string that we have transformed into 7 numeric attributes: size of the string, number of lowercase letters, capital letters, numbers For the detection of malicious PDF files, we have extracted 120 digital attributes, similar to those presented in the work of Smutz and Stavrou [19, 20], which we can group into three categories:

- file metadata (for example the author, or the date of creation);
- file structure (for example the number of objects, or the average size of the objects);
- objects and keywords used in the file (for example the types of objects, or the number of image objects).

C. Selection of the type of classification model

All the intelligence of supervised learning lies in optimizing the parameters of the classification model from the training data for which the labels are known. This optimization phase is theoretically complex, but the great popularity of machine learning has gradually simplified its use with the emergence of many dedicated libraries (scikit-learn in python, Spark, Mahout or Weka in java, or Vowpal Wabbit in C ++ for example), and solutions online like Google Cloud ML, Microsoft Azure or Amazon Machine Learning. This makes it easy to learn various classification models from learning data. Neural networks are so popular [8], especially thanks to their extraordinary results in computer vision, that the amalgamation between deep learning and machine learning is often made. Neural networks are only one type of classification model, and there are many others: decision trees, random forests (or Random Forests), k nearest neighbors, discriminating linear or quadratic analyzes, regressions logistics, support vector machines (or SVM for Support Vector Machine) or naive Bayesian classification to name a few. It is therefore necessary to choose a type of classification model suited to the detection problem considered, and meeting the operational constraints of the detection systems mentioned in section 2. Linear classification models, such as logistic regression or support vector machines, are adapted to intrusion detection and we now detail how they respond to operational constraints. Fast predictions Most classification models have a learning phase which is very costly in computation time and which is carried out line but applying the model to new data is usually extremely

fast. This is the case of linear classification models whose application is in O (d) where d is the number of attributes describing each instance. On the other hand, certain models, described as lazy, are to be avoided for intrusion detection. Indeed, they do not have a learning phase, and the entire learning data is therefore considered during the prediction phase. For example, the k nearest neighbors is a lazy model not suitable for intrusion detection. To predict the label of a new instance, you have to look for its k closest neighbors among all the training data, and then the predicted label is the one most represented. The predictions of this classification model have a time complexity in O (nd) which depends on the number of attributes d, but also on the number of training data n. However, in machine learning, it is desirable that n be large, and n increase over time when new instances are reinjected into the model following the analysis of the alerts generated. The prediction time of this classification model is far too long to be used in a detection system.

Periodic updating of the model When a detection system is in production, the security operator ensures its supervision by analyzing the alerts generated. Their primary purpose is to identify false alarms, but also to take the necessary action in the event of a security incident. However, their analyzes, both false and true positive, can be incorporated into the detection model to improve its detection capabilities. It is therefore essential that the model can be updated periodically without reconsidering the entirety of the training data used previously. Linear models perfectly meet this constraint. They can be learned in batch mode initially but can also be updated incrementally by taking into account only the new labeled data. Interpretable model Linear models have the great advantage of being interpretable. Thus, the administrator of the detection system can interpret the detection model, that is to say understand how he makes his decisions, before putting it into production in order to check its consistency. Then, the operator supervising the detection system can know the main causes responsible for generating an alert.

6. VALIDATION ON LEARNING DATA

We would like to point out that the performance of the detection model on its learning data is not a good evaluation of the model. Indeed, the goal of a detection model is not to correctly classify training data, but to be able to generalize, that is to say, to correctly classify data not used during its learning phase. However, analyzing the performance of the model on the training data can help detect the problem of under-learning. It is said that there is under-learning when the detection model has learned almost nothing from the training data. In this situation, the detection model behaves almost like a random generator. To diagnose under-learning, simply draw the ROC curve obtained on the training data and check that it is not too close to that of a random generator. The expert has two options to resolve the under-learning: add discriminating attributes or use a more complex type of classification model. When a model fails to discriminate between malicious and benign instances on learning data, it is often that the expert has not given good attributes as input. It is therefore necessary for it to resume the attribute extraction phase to add more discriminating characteristics. Sometimes the expert provided discriminating attributes, but the type of classification model chosen is not complex enough to discriminate against malicious instances of benign. Figure 6 (a) shows a two-dimensional dataset where a linear model is not complex enough to properly separate malicious instances from benign, whereas a quadratic model, slightly more complex, is perfectly suited (see Figure 6 (b)). However, using an extremely complex detection model is not a solution, because in this case we risk over learning. It is said that over-learning occurs when the classification model perfectly predicts the label of learning data, but is unable to correctly predict the label of new data. The over-learning problem can be diagnosed by analyzing the performance of the model on an independent validation set and is therefore presented in the following section.

7. CONCLUSION

In this article, we have described a methodology for solving the “black box” aspect often criticized by machine learning methods, and we illustrated it with a case study on the detection of malicious PDF files. Thanks to this methodology, it becomes possible to adapt machine learning to the reluctance of security experts, and to the constraints of intrusion detection systems. In particular, the article emphasizes the importance of the rigor of the model validation method before it goes into production. In fact, errors during the validation phase can overestimate the efficiency of a model, and therefore lead to unpleasant surprises during production. The validation methodology that we propose makes it possible to anticipate problems inherent in machine learning, and therefore to avoid additional costs linked to tests in production.

REFERENCES

[1] BARANDIARAN, X. Behavioral Adaptive Autonomy: a milestone on the ALife route to AI?. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL LIFE, ALIFE, 9th, 2004, Boston, MA, USA. Proceedings… Cambridge, MA: MIT Press, 2004. p.514-521.

[2] BARANDIARAN, X.; MORENO, A. On what makes certain dynamical systems cognitive. Adaptive Behavior, SAGE, v.14, n.2, p.171-185, 2006.

[3] BARANDIARAN, X.; MORENO, A. Adaptivity: From Metabolism to Behavior. Adaptive Behavior, SAGE, v.16, n.5, p.325-344, 2008.

[4] BEER, R.D. A dynamical systems perspective on agent-environment interactions. Artificial Intelligence, Elsevier, v.72, p.173-215, 1995.

[5] BEER, R.D. Autopoiesis and Cognition in the Game of Life. Artificial Life, MIT Press, v.10, p.10-309, 2004.

[6] BELLMAN, R. A Markovian Decision Process. Journal of Mathematics and Mechanics, Bloomington: Indiana University Press, v.6. p.679-684, 1957.