Journal of Machine Learning Research 10 (2009) 1387-1390 Submitted 8/08; Revised 2/09; Published 7/09

Model Monitor (M^2): Evaluating, Comparing, and Monitoring Models

Troy Raeder TRAEDER@CSE.ND.EDU
Nitesh V. Chawla NCHAWLA@CSE.ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA

Editor: Soeren Sonenberg

Abstract

This paper presents Model Monitor (M^2), a Java toolkit for robustly evaluating machine learning algorithms in the presence of changing data distributions. M^2 provides a simple and intuitive framework in which users can evaluate classifiers under hypothesized shifts in distribution and therefore determine the best model (or models) for their data under a number of potential scenarios. Additionally, M^2 is fully integrated with the WEKA machine learning environment, so that a variety of commodity classifiers can be used if desired.

Keywords: machine learning, open-source software, distribution shift, scenario analysis

1. Introduction

Most work in machine learning implicitly assumes a *stationary distribution*, meaning that the population from which our data is drawn does not change over time. However, a number of real-world applications present the challenge of a drift in distribution between training and testing (Quiñonero et al., 2008). At the same time, the prevalent evaluation paradigm does not take into account the effect of changing distributions on the performance of the models being compared. We posit that it is important to effectively evaluate the generalization capacity of models or classifiers on a range of distribution divergences. We propose M^2, a Java toolkit that is designed to help researchers and practitioners grapple with potential shifts in data distribution. The fundamental issues that we address in this toolkit are: a) which of several competing models is most robust to changing distributions? b) when, why, and how may the learned model fail? Specifically, it carries the following fundamental capabilities.

1. Detection of distribution shift: M^2 contains methods for isolating features whose distribution has changed significantly between training and testing sets. This allows for following: a) sensitivity/fragility analysis on features with respect to distributional shifts; b) if a testing set is available, determination of features that change significantly between the training and testing sets; and c) if an effective model is beginning to degrade in performance, problematic features can be isolated and the model can be updated before its performance gets worse.

2. Exploration of hypothetical scenarios: If a user is building a model on new data and expects the distribution to change over time, he or she can inject these changes into the test data and

determine which classifiers[1] (trained on current data) perform best on both the current and hypothetical future distributions.

3. Empirical comparison of classifiers: M^2 can compare classifiers on performance across several data sets and distribution shifts. This capability allows researchers to objectively evaluate the robustness of the classifiers under distribution shift. This can guide the selection of a classifier that is more robust to distributional drifts. Various cross-validation schemes are available, and we implement statistical tests, so that the user knows whether the observed differences are statistically significant.

M^2 is, to our knowledge, the only publicly-available software to tackle the important problem of learning under changing distributions. At present, it is designed mainly for two-class problems. Most of the currently-implemented performance measures will compare the performance of the positive class (i.e., class 1) against the performance on all other classes.

The current version is available on `mloss.org`, and its distribution package contains a user's guide, developer's guide, JavaDoc documentation, and examples. To afford flexibility to the user M^2 fully integrates with WEKA (Witten et al., 1999), but also provides support for interfacing with arbitrary custom classifiers. Input data can either be in the WEKA format or the comma-delimited format of traditional UCI repository data sets (often called C4.5 style).

2. Implementation

M^2 is implemented in Java and is therefore fully cross-platform. It provides a Swing graphical user interface, and implements several different distribution shifts, methods of evaluation such as 10-fold, 5x2 CV, etc., performance measures, and statistical measures for the comparison of classifiers. Each of these critical components has been validated by a series of automated unit tests to insure correct operation and protect against regression.

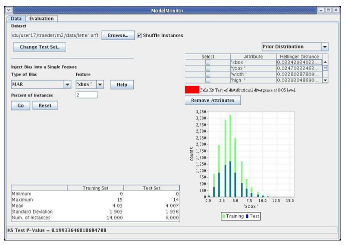
2.1 Performance Measures

In addition to the standard performance measures (*Accuracy*, *AUROC*,*Precision*, *Recall*, F_1-*Measure*), we have implemented other performance and loss measures commonly used for binary classification tasks, including *Brier Score* and *Negative Cross Entropy*. For any of the tasks described above, M^2 supports all of the standard validation measures including *cross-validation* (5-by-2 and 10-fold from the GUI, arbitrary configurations otherwise), splitting a data set into training and testing, and specifying an entirely separate file of test data.

2.2 Distribution Shifts

M^2 can introduce a number of different changes in distribution, including *missing at random*, *missing not at random*, *random noise*, *mean shift*, *variance change*, and a *prior probability shift*. Specific details on how each of the distribution shifts are implemented is available in the user's guide.

1. Model and classifier will be used inter-changeably in this paper.



edu/user17/traeder/m2/data/letter.arff

Type of Bias: MAR
Feature: 'xbox '
Percent of Instances: 2

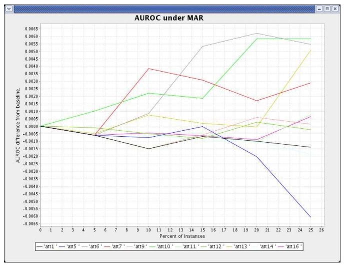
Select	Attribute	Hellinger Distance
	'xbox '	0.03342934023...
	'ybox '	0.02470332463...
	'width '	0.03280287809...
	'high '	0.03393048690...

🟥 Fails KS Test of distributional divergence at 0.05 level.

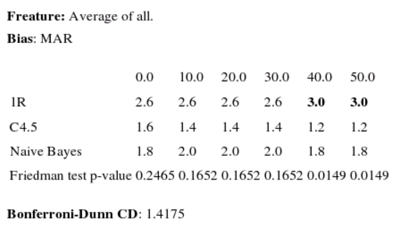
	Training Set	Test Set
Minimum	0	0
Maximum	15	14
Mean	4.03	4.007
Standard Deviation	1.903	1.936
Num. of Instances	14,000	6,000

KS Test P-Value = 0.19933646010684788

(a) Detecting distribution shift



(b) Plot of classifier sensitivity to MAR bias.



Freature: Average of all.

Bias: MAR

	0.0	10.0	20.0	30.0	40.0	50.0
1R	2.6	2.6	2.6	2.6	**3.0**	**3.0**
C4.5	1.6	1.4	1.4	1.4	1.2	1.2
Naive Bayes	1.8	2.0	2.0	2.0	1.8	1.8
Friedman test p-value	0.2465	0.1652	0.1652	0.1652	0.0149	0.0149

Bonferroni-Dunn CD: 1.4175

(c) Comparison of classifiers across multiple data sets.

Figure 1: Types of data presented to the user.

2.3 Statistical Measures

The bulk of the implementation has to do with methods for detecting and evaluating the effect of shifts in distribution. For detecting shifts in distribution, we provide both *Hellinger distance* and the *Kolmogorov-Smirnov* (KS) test. Recent work by Cieslak and Chawla (2007) has shown that systematic shifts in distribution tend to cause significant changes in Hellinger distance. The KS test helps to distinguish between changes caused by random fluctuation and changes caused by systematic bias.

For comparing the performance of multiple classifiers, we have implemented the *Friedman test* and the *Bonferroni-Dunn test*. The average performance of several classifiers, across multiple data sets is compared with the Friedman test, as suggested by Demšar (2006), and if a significant difference in performance is found, the Bonferroni-Dunn test determines the classifiers or sets of classifiers between which the significant difference exists.

3. Functionality

Here we provide a quick overview of the toolkit's major functionality. Each feature is described in greater detail in the user's guide.

3.1 Detecting Distribution Shift

When a user starts M^2, he or she can load any data set in either C4.5 names/data format or the WEKA ARFF format. The user then specifies a test set (either by selecting hold-out or cross-validation or by choosing a separate test file).

At this point, the user can begin evaluating the training and test data sets for shifts in distribution. Each attribute in the data set is automatically displayed, and selecting it will produce a histogram of both distributions so that any differences can be visually inspected. Additionally M^2 automatically calculates the Hellinger distance for each feature between the training and test sets. If a feature's distribution fails the KS test (at the $\alpha = 0.05$ confidence level), the Hellinger distance for that feature is highlighted. In this way, users can quickly locate features whose distribution may have changed. A simple example appears in Figure 1(a).

3.2 Exploration of Hypothetical Scenarios

After loading a data set, the user has the opportunity to introduce any of a number of distribution shifts into the test data. Upon doing so, the histograms displayed on the screen will be updated, and new Hellinger distances and KS-test p-values are calculated. The user can then quickly evaluate any classifier on both the original and new ("biased") test sets to compare the results. WEKA classifiers can be chosen from a GUI chooser, whereas for general classifiers, a commandline must be specified.

For more sophisticated sensitivity analysis, the user may specify a range of distribution shifts. In this case, he or she provides a set of increasingly severe biases, and then the tool evaluates the classifier's performance under each one. The results are presented to the user both graphically (plotting classifier preformance against severity of shift) and in tabular form, to facilitate further offline processing.

3.3 Empirical Comparison of Classifiers

Finally, users may test any one classifier against several others across multiple data sets and distribution shifts. For each specified distribution shift, the software will evaluate each classifier on all the data sets. It ranks the classifiers on each data set, calculates the average rank for each classifier, and then performs the Friedman test for analysis of variance on the resulting table of ranks. If the Friedman test determines (again at the 0.05 level) that there is a significant difference between the classifiers, we run the Bonferroni-Dunn test to determine exactly which classifiers are different from the specified reference classifier.

In addition to producing an annotated table of ranks containing the statistical test information, this step produces all the same output that the other portions of the program produce. Taken together, these results provide a complete picture of the classifiers' relative performance in the relevant scenarios.

Acknowledgments

The authors would like to acknowledge Ryan Lichtenwalter, Dave Cieslak, and Karsten Steinhaeuser for helpful discussions on both implementation and user interface issues.

References

D.A. Cieslak and N.V. Chawla. Detecting Fractures in Classifier Performance. *ICDM 2007*, pages 123–132, 2007.

J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *JMLR*, 7:1–30, 2006.

J. Quiñonero, M. Sugiama, A. Schwaighofer, and N. D. Lawrence, editors. *Dataset Shift in Machine Learning*. MIT Press, 2008.

I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S.J. Cunningham. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. *ICONIP/ANZIIS/ANNES*, 99:192–196, 1999.