Contents

Document history

Version	Date	Author (Unit)	Description
v0.3			The test and evaluation framework discussed and established
v0.6			Initial draft finished and reviewed by the PSRG
v1.0			Final version of document

1 Introduction

Even though the number of algorithms designed for learning on streaming data is increasing, there is still not a unified and well accepted way for evaluating them. This is because testing and evaluating algorithms that are designed to work on streaming data are more difficult those designed to work on static data. There are both statistical and computational reasons for this.

Static data is data, where each instance can be assumed to be identically and independently distributed i.i.d. On streaming data, one can often not assume that data instances are i.i.d. Moreover, the algorithms are often designed to weight measurements that are close to the actual time step higher than measurements that are further back. On streaming data, we must therefore assume that data are generated from underlying distributions that are time dependent and also that the algorithms themselves are time dependent.

Computational challenges are related to the fact that the data come from an open-ended data stream, conceptually infinitely long, which imposes practical challenges related to restrictions on cpu-time and memory allocation.

Various error measures related to stream data has been proposed in the papers of Gama et. al. [2], [3], [4]. A loss function is typically related to the penalty of misclassifications on classification problems or residuals in regression models. The holdout error is basically the average loss on a holdout dataset of fixed size. The predictive sequential, or *prequential* error is defined as the average loss function up to time step i, where i is the current time step. Moreover, it was also suggested to use a prequental error measure, which involved a forgetting factor such as using a time window or fading factors. In paper [4], convergence towards the Bayes error was shown for all these performance measures provided that the learners are consistent and data are i.i.d. Moreover, it was shown that if data was allowed to drift over time, meaning that samples are only locally i.i.d, then the prequental error measures with forgetting mechanisms were favourable.

[**TODO: Relate to other work on streaming data as well.**]

The applications that are covered in this report have different characteristics than the applications discussed in [4]. Some problem will be evaluated on a holdout dataset assuming i.i.d and a stationary algorithm. Other problems can not assume i.i.d on a local scale, but stationarity can be assumed on a larger time scale.

In this paper we will establish formal procedures for testing and evaluating the developed models and algorithms. This includes specification what metrics are relevant to use to quantify the ability of the AMIDST system, such as relevant formalization of loss functions, maximum response-times, memory limits and output format. The paper will also include considerations about what quantitative improvements AMIDST should obtain over state of the art.

In section 2, AMIDST relevant methodologies for evaluation of both batch and stream-

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Table 2.1: Example of a confusion matrix. A classifier is labelling instances as either cats, dogs or rabbits. The accuracy is the sum of the diagonal elements divided by the total number (in this case 19/27).

ing algorithms are identified and discussed. This section forms the foundation of the subsequent sections, where the exact evaluation routines for each use case provider is given. These sections contains a description of the requirements related to evaluation as described in Delivery 1.2, a short description of the algorithms and the data and finally methods for evaluating predictive and runtime performances. Section 6 concludes the report.

2 Test and evaluation methodology

As discussed in the introduction, finding appropriate performance measures on streaming data is more difficult than finding performance measures on static data. We will therefore start by discussing some relevant performance measures on static data. A brief section of state of the art methods on streaming data is included is before a justification of our approach is exposed.

2.1 Performance measures for classification on static data

On static data, one can assume that we have a dataset of fixed size n, where each instance is independently drawn from a joint probability distribution $P(X, Y)$, where X and Y are random variables. The X variable is known as the explanatory variable and Y is the class label. These two variables have output spaces Ω_X and Ω_Y, respectively. For instance in a binary classification problem Ω_Y can either be true or false, while Ω_X is a space of all possible explanatory vectors.

In classification, we typically consider a hypothesis function $h\Omega_X \to \Omega_Y$. In terms of evaluating the performance of h, we use a dataset of n input-output pairs (x_i, y_i), independently drawn from $P(X, Y)$. The result of such an experiment can be shown in a confusion matrix. An example is shown table 2.1, there the classifier is trying to distinguish cats, dogs and rabbits.

In this example one can see that it is easy to distinguish cats from rabbits and vice versa, while it is much harder to distinguish cats from dogs and vice versa. A more global

		Predicted class	
		Cat	Not cat
Actual class	Cat	5	3
	Not cat	2	17

Table 2.2: Example of a confusion matrix for a classifier of cats and not cats. The accuracy is 22/27.

		Actual Condition	
		Positive	Negative
Test outcome	Positive	5	2
	Negative	3	17

Table 2.3: Example of a confusion table for a classifier of cats and not cats. The true positives and true negatives are on the diagonal, while the two other numbers are the false positives and the false negatives.

measure of the classification algorithm is the classification accuracy which is basically the diagonal elements divided by the total number (in this case 19/27). It is important to note that the accuracy is not telling the whole story of the classification rule. For instance, when there are a lot more instances of one class compared to the others, a naive classification rule that always predict the majority class will get a high accuracy, even though the method is not using the information in the explanatory variables at all. This is why we need to show the full matrix to interpret the status of the classification rule.

When the classification is a binary, such as classifying cats and non-cats, the confusion matrix becomes two dimensional as shown in 2.2. In binary classification, it is common to introduce positives and negatives, instead of the real class names. A true positive is therefore an actual cat that has been predicted to be a cat by a classifier. False positives, true negative and false negatives is defined in a similar manner. The results are commonly shown in confusion tables, such as table 2.3.

From a confusion table it is easy to calculate numerous numbers that describes the classification rule. We mention here the true positive rates, also known as recall, which is the number of true positives divided by the total number actual positives. Another important value is the false positive rates, also known as fall-out, which is the number of false positives divided by the total number actual positives.

[TODO: Possibly add a bit more about other measures based on the confusion table, that could be considered.]

By investigating various numbers that can be deduced from the confusion table it is possible to discuss classification rules when the datasets are highly skewed as well. However, these numbers do not take into account that some misclassifications might be more costly than others. On one side, a false positive might be more costly than a false negative. For instance, if we are detecting cancer on patients it might be more costly to not detect a sick person than saying that a not sick patient is sick. Moreover, the cost of each false positives may not be constant either. For instance, if the classifier is predicting whether a client in a bank will default a loan or not, the cost is clearly related to the size of that loan. The next subsection includes a procedure to include such costs in the performance measures.

2.1.1 Empirical risk

[TODO: Go over the notation again.]

In mathematical optimization, statistics, decision theory and machine learning, a loss function or cost function is a function that maps an event or values of one or more variables onto a real number that is intuitively representing some *cost* associated with the event. Loss functions can be used on optimization problems, where an algorithm or method is optimized by minimizing the loss function. Moreover, loss functions are frequently used to diagnose and compare various algorithms or methods.

In this paper define the *loss function* as a real and lower-bounded function L on $\Omega_X \times \Omega_Y \times \Omega_Y$. The value of the loss function at an arbitrary point $(x, h(x), y)$ is interpreted as the loss, or cost, of taking the decision $h(x)$ at x, when the right decision is y. Notice that in this paper, the loss function is dependent on x as well. This is of high practical use, because a certain misclassification might be more expensive than another.

In the frequentist perspective, the expected loss is often referred to as the risk function. It is obtained by taking the expected value over the loss function with respect to the probability distribution $P(X,Y) : \Omega_X \times \Omega_Y \to \mathbb{R}^+$. The *risk function* is given by

$$R(h) = \int_{\Omega_X,\Omega_Y} L(x, h(x), y) dP(x, y). \tag{2.1}$$

In the case when the costs are independent of x and also that there is no cost related to correct classification, the risk function reduces to the well known expected cost of misclassification (ECM)

$$ECM = c(1|0)p(1|0)p_0 + c(0|1)p(0|1)p_1. \tag{2.2}$$

Here, $c(1|0)$ is the cost for misclassifying an item of class zero as class one and $p(1|0)$ is the misclassification probability given class zero. The quantities $c(0|1)$ and $p(0|1)$ are defined equivalently, while p_0 and p_1 are the priors.

In general, the risk $R(h)$ cannot be computed because the distribution $P(x, y)$ is unknown. However, we can compute an approximation, called empirical risk, by averaging the loss function on the training set given by

$$R_{emp}(h, \mathbf{x}) = n^{-1} \sum_{i=1}^{n} L(x_i, h(x_i), y_i). \tag{2.3}$$

Notice that L is an array of $n \times 2 \times 2$ elements. Many supervised learning algorithms are optimized by finding the h in a hypothesis space $\mathcal{H}$ that minimizes the empirical risk. This paper will not focus on empirical risk minimization, but rather focus on using empirical risk to compare methods.

2.1.2 Evaluation of families of classification rules

So far we have discussed how to evaluate a single classification rule. However, most classification rules in the AMIDST framework is based on comparing an estimated probability to a certain threshold. We call this estimated probability the output function $q : \Omega_X \to \mathbb{R}^+$. The classification rules are the family of hypothesis functions $\mathcal{H}$, where each element $h_T : \Omega_X \to \Omega_Y$ has the form

$$h_T(x) = \begin{cases} 0 & \text{for} \quad q(x) \leq T \\ 1 & \text{else.} \end{cases} \tag{2.4}$$

It is of interest to evaluate all these classification rules. The receiver operating characteristic ROC is a plot of true positive rate as a function of false positive rate, as T is allowed to vary over all relevant variables. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making.

[TODO: Add a ROC plot.]

[TODO: Improve the Relation to the Mann-Whitney U test below]

Also, let X_0 and X_1 be random variables with probability distributions $P(X|Y=0)$ and $P(X|Y=1)$, respectively. We define the random variables $Q_0 = q(X_0)$ and $Q_1 = q(X_1)$.

In this paper, we will discuss the continuous output function in the light of the Mann-Whitney U test and the concordance probability $P(Q_1 > Q_0)$. It is important to mention that in the context of concordance probability, the shapes of $P(Q_0)$ and $P(Q_1)$ may be different.

Also, we want to mention that the concordance probability is exactly equal to the area under the receiver operating characteristic curve (ROC) and the common language effect size of the Mann-Whitney U test.

In terms of discussing the family of hypothesis functions, we have chosen empirical risk as the quantity of interest. This involves defining a loss function and the risk function is simply the expected loss in a frequentist perspective.

2.1.3 Mann-Whitney U test

In statistics, the Mann-Whitney U test (also called the Mann-Whitney-Wilcoxon (MWW), Wilcoxon rank-sum test, or Wilcoxon-Mann-Whitney test) is a nonparametric test of the null hypothesis that two populations are the same against an alternative hypothesis, especially that a particular population tends to have larger values than the other. It has greater efficiency than the t-test on non-normal distributions and it is nearly as efficient as the t-test on normal distributions.

We define a training set $\mathbf{x} \times \mathbf{y}$ with n input-output pairs (x_i, y_i), independently drawn from $P(X, Y)$. From the training set we have two populations $\mathbf{q_0} = \{q(x_i), | \, y_i = 0\}$ and $\mathbf{q_1} = \{q(x_i), | \, y_i = 1\}$. Their sizes are n_0 and n_1 so that $n_0 + n_1 = n$. Calculating the U statistics is straightforward, where these two values are obtained

$$U_0 = \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} H(\, q_j - q_i \,) \quad \text{and} \quad U_1 = \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} H(\, q_i - q_j \,). \tag{2.5}$$

Here $H(\cdot)$ is the heaviside step function and notice that $U_0 + U_1 = n_0 n_1$. For large samples, each U is approximately normally distributed. In that case, the standardized value

$$z = \frac{U_0 - m_U}{\sigma_U}, \tag{2.6}$$

where m_U and σ_U are the mean and standard deviation of U given by

$$m_U = \frac{n_0 n_1}{2} \quad \text{and} \quad \sigma_U = \sqrt{\frac{n_0 n_1 (n_0 + n_1 + 1)}{12}}. \tag{2.7}$$

Significance of test can be checked in tables of the normal distribution. Although, such an hypothesis test is interesting by itself, we are more interested in the concordance probability $P(Q_1 > Q_0)$ which is defined by

$$P(Q_1 > Q_0) = \frac{U_1}{n_0 n_1}. \tag{2.8}$$

3 Cajamar: test and evaluation

3.1 Use case requirements

Summarize the use case requirements for the different application scenarios. This information should be derived from Deliverable 1.2.

3.2 Model and data characteristics

Describe aspects of the model and data relevant for the ensuing test and evaluation discussion. Much of this information can be synthesized from the existing documents, and should serve to make the document more self-contained.

3.3 Predictive performance: test and evaluation

3.3.1 Application scenario 1

3.3.2 Application scenario 2

3.4 Run-time performance: test and evaluation

3.4.1 Application scenario 1

3.4.2 Application scenario 2

4 Daimler: test and evaluation

4.1 Use case requirements

Summarize the use case requirements for the different application scenarios. This information should be derived from Deliverable 1.2.

4.2 Model and data characteristics

Describe aspects of the model and data relevant for the ensuing test and evaluation discussion. Much of this information can be synthesized from the existing documents, and should serve to make the document more self-contained.

4.3 Predictive performance: test and evaluation

4.3.1 Application scenario 1

4.3.2 Application scenario 2

4.4 Run-time performance: test and evaluation

4.4.1 Application scenario 1

4.4.2 Application scenario 2

5 Verdande: test and evaluation

5.1 Use case requirements

Summarize the use case requirements for the different application scenarios. This information should be derived from Deliverable 1.2.

5.2 Model and data characteristics

Describe aspects of the model and data relevant for the ensuing test and evaluation discussion. Much of this information can be synthesized from the existing documents, and should serve to make the document more self-contained.

5.3 Predictive performance: test and evaluation

5.3.1 Application scenario 1

5.3.2 Application scenario 2

5.4 Run-time performance: test and evaluation

5.4.1 Application scenario 1

5.4.2 Application scenario 2

6 Conclusion

References

[1] Kaptein, M.: Rstorm: Developing and testing streaming algorithms in r. The R Journal **6**(1) (2014) 123–132

[2] Gama, J.a., Rodrigues, P.P., Sebastião, R.: Evaluating algorithms that learn from data streams. In: Proceedings of the 2009 ACM Symposium on Applied Computing. SAC '09, New York, NY, USA, ACM (2009) 1496–1500

[3] Gama, J.a., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2009) 329–337

[4] Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine Learning **90**(3) (2012) 317–346