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 generally more complicated than those designed to work on non streaming data. There are both statistical and computational reasons for this.

If the data instances in a data stream are identically and independently distributed i.i.d., then the only new challenge compared to static i.i.d. data is that the data streams are open ended. It can be seen as a static i.i.d data that continuously grow in size. Evaluation methods related to these streams are closely related to the evaluation methods that are known for static i.i.d. data.

A generalization of i.i.d. data is stationary data, which is data that is generated from a stationary process. A stationary process is a stochastic process where the joint probability distribution over any sized time window do not change when shifted in time. Consequently, parameters such as the mean and variance do not change over time.

Various performance measures on stationary data streams have been proposed in the literature. Performance measures involving loss functions have been proposed in the papers of Gama et. al. [1], [2], [3]. The loss function on regression problems is a function of both the predicted value and the ground truth, while the loss function on classification problems is a function of predicted and real class labels. 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.

In this paper, we focus the exposition on binary classification, although we are aware that much can be generalized to multi classification or regression. Moreover, most classification rules in the Amidst software involves comparing an output function, or a score, to a threshold. The output function in the Amidst software is usually a Bayesian network that predicts a probability of a class label. In this case, the area under the receiver operator characteristics curve (AUC) is an interesting alternative for stationary streams.

AUC is a popular method for evaluating classification problems where class imbalance is vital, because it is invariant to the class distribution. In the case of i.i.d. data, AUC has the statistical interpretation that it is the probability that a member of the "positive" class is scored higher than a member of the "negative" class. AUC is therefore a measure of the *ranking ability* of the output function. This is particularly relevant if one wants to change the classification threshold as a consequence of changing class distributions or changing misclassification costs [7]. Moreover, it also pointed out that AUC is more preferable than accuracy for model evaluation in [8].

Specialized learning algorithms on imbalanced streams are proposed in [4–6], where these papers points out that this problem is particularly difficult and effective algorithms for

evaluating such a classifier is vital. In the papers [4, 6], the area under the receiver operator characteristics curve (AUC) is calculated on limited holdout sets, while in [5] AUC is calculated on the entire stream.

AUC calculation involves sorting all instances and iteration over the sorted list. In a streaming context, this means that the calculation is $O(n)$ where n is the length of the data stream. If the hole stream is used, it may have computational problems related to memory and cpu time.

However, most streams are not stationary. Problems that involve *consept-drift* are problems when both data are drifting and the learners are changing over time. In [3], it was suggested to use prequental loss with forgetting mechanisms to compensate for concept drift. The forgetting mechanisms involves either using a time window or fading factors. In paper [3], convergence towards the Bayes error is shown for all these performance measures provided that the learners are consistent, loss is zero-one and data is i.i.d. Moreover, it is shown that in the case of concept-drift, the prequental error measures with forgetting mechanisms are favorable over those without a forgetting mechanism.

In [9], the prequential AUC with a forgetting mechanism is proposed. On one side this measure allows for concept drift and on the other side it solve the computational problems in [4–6] as pointed out above.

In the AMIDST project there are use case scenarios where the data is multiple i.i.d. streams. These use case scenarios are dealt with by dividing these streams into sub streams over a particular time window, where a evaluation time is in between the start time and the end time. The data from the start time to the evaluation time is basically the explanatory variables and the data at the end time is basically the class label. This problem reduces to a problem of non streaming nature, and the evaluation is not taking concept-drift into account. However, it is worth noting that this problem can be expanded in the sense of prequental loss as of [3] or prequental AUC as of [9].

Moreover, there are also use case scenarios in the Amidst project that do not directly fit the prequental performance measures as of [3] and [9]. All of these measures involve calculating a loss or a score at each time step. In this report, we will describe a use case scenario where it is important to predict that certain event is happening somewhere inside an *interval* prior to the event actually takes place.

This problem has been solved by chopping the streams into a number of sub streams that are approximately i.i.d and labeled as either a positive or a negative. It is important that enough space between sub streams are left so that dependencies are withdrawn. In essence, the i.i.d. assumption of the sub streams allows one to investigate the output function independently from the prior distributions and the choice of loss functions.

In section 2, AMIDST relevant methodologies for evaluation of both batch and streaming algorithms are identified and discussed. This section forms the foundation of the next three 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

		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).

described in Delivery 1.2 and a short description of the algorithms and the data. At the end of these sections, the methods for evaluating predictive and runtime performances are exposed and discussed. 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 non streaming data. We will therefore start by discussing some relevant performance measures for classification on non streaming data that is i.i.d.

2.1 Performance measures for classification on i.i.d. data

We assume that the dataset has a 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) that are independently drawn from $X \times Y$. The result of such an experiment can be shown in a confusion matrix. An example is shown table 2.1, where the classifier is attempting to distinguish cats, dogs and rabbits.

A global 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, by looking at the table it is seen that the algorithm distinguishes cats from rabbits quite easily, but it is much harder to distinguish cats from dogs. This information can not be found by only looking at accuracy.

		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.

Moreover, accuracy is also very dependent on how evenly the classes are distributed. 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 any of the information that is contained in the explanatory variables. These are reasons for inspecting the full confusion matrix for interpreting the classification rule. There are also more numbers that can be derived from the confusion matrix, but to simplify the exposition, we have chosen to limit the discussion to binary classification.

For a binary classifier, such as classifying cats and non-cats, the confusion matrix is two dimensional as shown in table 2.2. In binary classification, it is common to introduce positives and negatives, instead of the class labels. A true positive is therefore an actual cat that has been predicted to be a cat. False positives, true negative and false negatives are defined in an equivalent manner. The results are commonly shown in a confusion table (see table 2.3), which is not the same as a confusion matrix.

From a confusion table it is easy to calculate numerous numbers that describes the classification rule. Specifically, we mention the true positive and false positive rates. True positive rates, also known as recall, is the number of true positives divided by the total number actual positives. The false positive rates, also known as fall-out, is the number of false positives divided by the total number actual positives.

By investigating various numbers that can be deduced from the confusion table, it is possible to discuss classification rules, even when the datasets are unevenly distributed. That is, when some class label have more instances than others.

However, these numbers do not take into account that some misclassifications might be more costly than others. For instance, a false positive might be more costly than

a false negative, such as in the case of cancer diagnostics. It might be more costly to not treat a person that is sick, compared to treating a healthy person. Moreover, the cost of each false positive (or false negative) 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 the loan in question. The next subsection includes a procedure to include such costs in a performance measure.

2.1.1 Empirical risk

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, we 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 data set Z of size n, where each element is (x_i, y_i). The empirical risk is given by

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

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 the 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 $q : \Omega_X \to \mathbb{R}^+$. In the AMIDST framework the range of q is usually $[0, 1]$, but this restriction is not necessary for this theory. The potential 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 the true positive rate as a function of the false positive rate, as T 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.

The ROC curve allows a visual exposition of the family of classifiers $\mathcal{H}$. In particular, it is easy to see which threshold is needed for a certain hit rate or true positive rate. Moreover, it is also of interest to reduce all this information into a single number. The area under the ROC curve, also known as AUC, is such a number. AUC is independent of T and also the priors. It may be difficult to comprehend what AUC is at this point, but we will return to it afterward.

First, 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 $Q_1 = q(X_1)$. To summarize, Q_0 is basically the output value you get when you pick a random sample of class zero. The same thing can be said for Q_1.

The probability $P(Q_1 > Q_0)$ is of interest, because this says what is the probability that if you take one sample from each of the populations, what is the chance that the output value from sample one is higher than the output value of sample zero. This question is independent of T, meaning that the discussion about priors and costs are not needed. This probability is called the concordance probability.

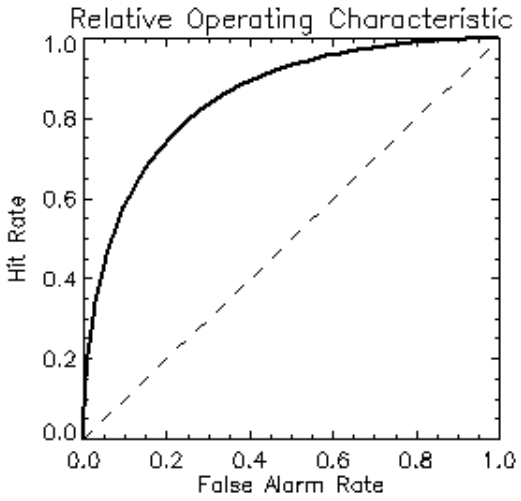

Figure 2.1: Receiver operating characteristics curve. True positive or hit rate is shown as a function of false positive or false alarms rate. The dashed line shows the ROC curve of the random guess.

Without exposing the details, it is possible to show that in the case that the data set is i.i.d.

$$\text{AUC} = P(Q_1 > Q_0). \tag{2.5}$$

It is important to note that nothing need to be assumed about the probability distributions of Q_0 and Q_1. Furthermore, it is worth noting that the concordance probability is exactly equal the common language effect size of the Mann-Whitney U test. We have therefore chosen to add a section of the Mann-Whitney U test.

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 data set Z with n input-output pairs (x_i, y_i), independently drawn from $P(X, Y)$. From the data 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

$$U_0 = \sum_{i=1}^{n_0}\sum_{j=1}^{n_1} H(\, q_j - q_i \,) \quad \text{and} \quad U_1 = \sum_{i=1}^{n_0}\sum_{j=1}^{n_1} H(\, q_i - q_j \,). \tag{2.6}$$

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

$$z = \frac{U_0 - m_U}{\sigma_U}, \tag{2.7}$$

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

$$m_U = \frac{n_0 n_1}{2} \quad \text{and} \quad \sigma_U = \sqrt{\frac{n_0 n_1 (n_0 + n_1 + 1)}{12}}. \tag{2.8}$$

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

$$P(Q_1 > Q_0) = \frac{U_1}{n_0 n_1}. \tag{2.9}$$

It is important to note that even though the Mann-Whitney U test is dependent on wether the shapes of the probability distributions of Q_0 and Q_1 are similar, this requirement is not necessary for the concordance probability. Also. equation (2.9) shows a simple way of calculating $P(Q_1 > Q_0)$ and AUC.

2.2 Performance measures for classification on streaming data

This section includes various evaluation methods for classification on data streams. Central to choices of evaluation methods are stationarity versus consept-drift, class imbalance, wether a scoring function exist, computational constraints and also the importance of earliness of warnings.

2.2.1 Stationary data streams

Definition of stationarity. Prequential loss function Prequential AUC Computational constraints

2.2.2 Data streams that involve concept-drift

More details on concept-drift Prequential loss function with forgetting mechanism Prequential AUC with forgetting mechanism

2.2.3 Division into i.i.d. substreams

Statistical interpretability (The concordance probability)

2.2.4 Early and late warnings

Methods or reflections on how to incorporate The substream method

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] 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

[2] Gama, J., 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

[3] Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine Learning **90**(3) (2013) 317–346

[4] Ditzler, G., Polikar, R.: Incremental learning of concept drift from streaming imbalanced data. IEEE Transactions of Knowledge and Data Engineering **25**(10) (2013) 2283–2301

[5] Hoens, T., Chawla, N.: Learning in non-stationary environments with class imbalance. In: 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2012) 168–176

[6] Lichtenwalter, R., Chawla, N.: Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In: New Frontiers in Applied Data Mining. Volume 5669 of Lecture Notes in Computer Science. (2010) 53–75

[7] Wu, S., Flach, P., Ramirez, C.: An improved model selection heuristic for auc. In: ECML PKDD. Volume 4701 of Lecture Notes in Computer Science. (2007) 478–489

[8] Gama, J.: Knowledge Discovery from Data Streams. Chapman and Hall (2010)

[9] Brzezinski, D., Stefanowski, J.: Prequential auc for classifier evaluation and drift detection in evolving data streams. In: Proceedings of the 3rd International Workshop on New Frontiers in Mining Complex Patterns, Nancy, France, September 19, 2014. (2014)