Co-training semi-supervised learning for single-target regression in data streams using AMRules

Ricardo Sousa[1] and João Gama[1,2]

[1] LIAAD/INESC TEC, Universidade do Porto, Portugal
rtsousa@inesctec.pt
[2] Faculdade de Economia, Universidade do Porto, Portugal
jgama@fep.up.pt

Abstract. In a single-target regression context, some important systems based on data streaming produce huge quantities of unlabeled data (without output value), of which label assignment may be impossible, time consuming or expensive. Semi-supervised methods, that include the co-training approach, were proposed to use the input information of the unlabeled examples in the improvement of models and predictions. In the literature, the co-training methods are essentially applied to classification and operate in batch mode.
Due to these facts, this work proposes a co-training online algorithm for single-target regression to perform model improvement with unlabeled data. This work is also the first-step for the development of online multi-target regressor that create models for multiple outputs simultaneously. The experimental framework compared the performance of this method, when it rejects unalabeled data and when it uses unlabeled data with different parametrization in the training.
The results suggest that the co-training method regressor predicts better when a portion of unlabeled examples is used. However, the prediction improvements are relatively small.

Keywords: $Single-Target\ Regression \cdot Semi-Supervised\ Learning \cdot Co-training \cdot Data\ Streams$

1 Introduction

The importance of prediction has increased in online data streams context [1, 2]. In fact, several domains (where data is obtained through data streams) rely on the ability of making accurate predictions for decision making, planning, strategy development and reserve determination which depend on models produced by data analysis [3].

In this context, data streams produce massive quantities of data of which label assignment may be impossible, time consuming or expensive. Unlabeled data (without output values) is usually present in sensor malfunction or database

failure. In addition, labels may be omitted when data is sensitive (e.g, privacy preservation) or may be not obtained due to labeling cost [4]. Unlabeled data usually appear in a wide range of contexts such as Engineering Systems (video object detection) [5], Physics (weather forecasting and ecological models) [6], Biology (model of cellular processes) [7] and Economy/Finance (stock price forecasting) [3]. In most of these areas, data from streams are obtained and processed in real time [4].

Semi-supervised Learning (SSL) methodology has been developed to utilize the input information for accuracy improvement of the regression model by artificial labeling [4]. These methodologies become useful when the unlabeled data is significantly more abundant than labeled data [8]. However, these methodologies may introduce errors by propagating the inaccurate artificial labels [8].

Formally, let $\mathcal{S} = \{..., (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_i, y_i), ...\}$ denote an unlimited stream of data examples, where $\mathbf{x}_i = [x_{i,1} \cdots x_{i,j} \cdots x_{i,M}]$ is a vector of descriptive variables and y_i is a scalar output variable (label) of the i^{th} example (considering one example with the index of reference). The unlabeled example is represented with an empty label $y_i = \emptyset$. The aim of SSL consists of using examples $(\mathbf{x}_i, \emptyset)$ to improve the regression model $y_i \leftarrow f(\mathbf{x})$ and reduce the mean error of prediction for both labeled and unlabeled examples. Most of existent SSL methodologies are performed in batch mode with large amount of computational resources [2]. Moreover, these methodologies are often applied to classification and cannot be directly applied to regression [8].

Co-training has been showing promising results, among the SSL methods [4]. This method consists of creating two or more diversified models by using different input variables, different regressors or the same regressors with different parametrization. In the training stage, the regressors yield predictions that are processed (e.g., mean of all predictions or selection of the best prediction according to a criterion) to produce an artificial label in order to be used in the training of models. In the prediction stage, the regressors also yield predictions that are combined (e.g., weighted mean)to produce a final prediction.

The main propose of this paper is to apply and adapt the co-training method to online single-target regression context. This work also prospects the extension to online multi-target regression using the (Adaptive Model Rules)AMRules algorithm [2].

This paper is organized as follows. Section 2 presents a brief review on the principles of SSL and methods of co-training. Section 3 describes the adaptation of the co-training to online learning and regression. Section 4 explains in detail the evaluation method. Finally, the results are presented and discussed in Section 5 and the main conclusions are summarised in Section 6.

2 Related work

In this section, the main principles of co-training and some existent co-training methods are briefly reviewed. Since no online version of co-training methods were found in the literature, the most prominent co-training batch mode methods

are presented. Despite these facts, the methods are fair starting points for the development of online regression methods since they exhibit promising principles and results.

Co-training basically consists of two or more models training but with different aspects that allows to create diversity (different inputs, different regressors, different parametrization, ...). The common trait is that a regressor algorithm is trained with examples (previously unlabeled) artificially labeled by other complementary regressors. These regressors are assumed to predict reliably which makes the co-training confidence driven. This method relies on several assumptions such as consensus, complementary, sufficiency, compatibility and conditional independence.

- **Consensus** assumption states that the error reduction of labeled examples prediction and the increase of the unlabeled example prediction agreement lead to more precise models [10].
- **Complementarity** assumption states that each input group contains information that the other groups do not contain. Hence, the use of multiple input groups increase the amount of information to construct more accurate models [10].
- **Sufficiency** assumption considers that each group of inputs is adequate to build a model by proper training.
- **Compatibility** assumption implies that the output predictions from the models are very similar with high probability for the simultaneous input values of the respective groups.
- **Conditional independence** assumption allows the possibility of at least one of the model to produce less errors and teach the other models the correct prediction [11]. This assumption is essential for co-training, however it is very strong. To overcome this problem, similar but less demanding assumptions were consider. **Weak dependence** assumption, where some dependence exists between inputs was proven to work [12]. **Large diversity** assumption considers that independence can be achieved by using different algorithms or the same algorithm with different parametrization [13].

The main drawbacks of co-training are related to the inaccuracy of the artificially labeled examples that convey error to the models. In addiction, the artificially labeled examples may not carry the needed information to the regressor [8]. The co-training variants may present different strategies to artificially label the unlabeled examples or may present a criterion to discard the damaging artificially labeled examples. The prediction function generally combines the predictions of the models according to a criterion to produce the final prediction [8].

In this work the Co-training regression (COREG), Co-training by Committee for Regression (CoBCReg) and Co-regularised least squares regression (coRLSR) were studied. COREG uses two k-Nearest Neighbours (kNN) regressors [4]. Initially, the labeled and unlabeled examples are separated in two sets. For each regressor, the (k-NN) is used to construct a set of labeled examples which input

vector is close to the input vector of the unlabeled example by using a distance metric (user defined). Each regressor predicts a value to artificially label the example and uses it to re-train the models with all labeled examples. Mean Squared Error (MSE) variation is computed between the scenarios with and without the artificially labeled examples. If MSE is reduced, the artificially labeled example is joined to the labeled examples set. The process stops when none of unlabeled examples is interchanged between labeled and unlabeled sets. The final prediction is obtained by averaging the predictions of the two regressors. CoBCReg is based on Radial Basis Functions regressors (with a Gaussian basis function that uses the Minkowski distance) and Bagging. This algorithm implies that diversity must exist between the elements of the ensemble of regressors, which is achieved by different input subsets random initialization. In this method, each regressor selects the unlabeled examples that are more relevant for the respective model [14]. coRLSR (Co-regularised least squares regression) formulates into a regularised risk in Hilbert spaces minimisation problem [15]. It aims to find the models that minimizes the error of all models and the disagreement on unlabeled examples predictions.

3 Online Co-training regression

This section presents the proposed co-training method by showing the main adaptations to the online and regression context. This section also presents a small description of the underlying algorithm regressor AMRules.

The proposed co-training method, at the initialization stage, divides randomly the input variables of the incoming example into two groups and produce two example types with different input variables but with equal labels (labels of the incoming example). In this step, weak dependence is assumed. The two groups may overlap some inputs randomly selected by a pre-defined overlap percentage. Posteriorly, two AMRules complementary regressors yield predictions for each examples. The initial models are obtained previously in a training stage using a small dataset. Considering an incoming unlabeled example, a score is computed in order to evaluate the benefit/confidence of the prediction to be used in the artificial labeling for models training. The score is the relative error compared to maximum absolute value of the output found in the stream. If the score is lower than a pre-defined threshold, the predictions are used to train the complementary regressor. Otherwise, the artificially labeled example is discarded. The consensus assumption is used in this step. If the example is labeled, the mean error is computed for each regressor and the example is used for both regressor training. Algorithm 1 explains the training procedure of the proposed method.

For prediction, combination of the regressors are made by using weights. These weights are inversely proportional to the error produced by labeled examples previously used in the training stage. This strategy gives more credit to the regressor that produces less errors. Algorithm 2 explains the procedure of label prediction.

Algorithm 1 Co-training algorithm training

Initialization:
  $\alpha - Overlap\ percentage \quad s - Score\ Threshold$
  Random input allocation and overlapping into the two groups using α
Input: *Example* $(\mathbf{x}_i, y_i) \in \mathcal{S}$
Output: *Updated Models*
Divide $\mathbf{x}_i$ *into* $\mathbf{x}_i^1$ *and* $\mathbf{x}_i^2$
$\hat{y}_i^1 = PredictModel1(\mathbf{x}_i^1);\ \hat{y}_i^2 = PredictModel2(\mathbf{x}_i^2)$
if $(y_i = \emptyset)$ **then**
  if $(|\hat{y}_i^1 - \hat{y}_i^2|/|y_{max}| < s)$ **then**
    $TrainModel1((\mathbf{x}_i^1, \hat{y}_i^2));\ TrainModel2((\mathbf{x}_i^2, \hat{y}_i^1));$
else
  $\bar{e}_1 = Update\ the\ mean\ error\ of\ Model1(\hat{y}_i^1, y_i)$
  $\bar{e}_2 = Update\ the\ mean\ error\ of\ Model2(\hat{y}_i^2, y_i)$
  $TrainModel1((\mathbf{x}_i^1, \hat{y}_i^2));\ TrainModel2((\mathbf{x}_i^2, \hat{y}_i^1))$

Algorithm 2 Co-training algorithm prediction

Input: *Example* $(\mathbf{x}_i, y_i) \in \mathcal{S}$
Output: *Example prediction* $\hat{y}_i$
Divide $\mathbf{x}_i$ *into* $\mathbf{x}_i^1$ *and* $\mathbf{x}_i^2$
$\hat{y}_i^1 = PredictModel1(\mathbf{x}_i^1);\ \hat{y}_i^2 = PredictModel2(\mathbf{x}_i^2)$
$w_1 = \bar{e}_2/(\bar{e}_1 + \bar{e}_2);\ w_2 = \bar{e}_1/(\bar{e}_1 + \bar{e}_2);$
$\hat{y}_i = w_1 * \hat{y}_i^1 + w_2 * \hat{y}_i^2$

The AMRules regressor was used as the underlying algorithm in the training of the models and for output prediction of the unlabeled examples. The AMRules is a multi-target algorithm (predicts several outputs) that is based on rule learning[2]. AMRules partionates the input space and creates local models for each partition. The local models are trained using a single layer perceptron. Its main advantages are models simplicity, low computational cost and low error rates [2]. This algorithm presents convenient properties such as the modularity property that allows the construction of models for particular input variables regions (defined by the rule). It uses anomaly and change detection to increase resilience to data outliers and data changes on the stream. AMRules algorithm benefits from unlabeled examples since it prunes the input partitions.

4 The evaluation method

The proposed co-training algorithm was evaluated by simulating a data stream with artificial and real datasets. A percentage of 30 % of each dataset first examples (30 % of the first examples of the stream) were used to create an initial consistent model and the remaining examples were used in the testing stage.

In the testing stage, a binary Bernoulli random process with a probability p was applied to assign an example as labeled or unlabeled. If the example is

assigned as unlabeled, the true output value is omitted from the algorithm. The p probabilities of being unlabeled were 50%, 80%, 90%, 95% and 99%. Considering the algorithm parametrization, the score threshold values were $1\times10^{-4,}$, 5×10^{-4} 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 and 1. The diversity of these values allow to observe the behaviour of the algorithm in different scales. The overlap percentage values were 0%, 10%, 30%, 50%, 70% and 90%. Prequential mode was used in evaluation. This mode first predicts the label and then train the model for both labeled and artificially labeled examples [16].

Five real world and four artificial datasets were used. The real world datasets were House8L (Housing Data Set), House16L (Housing Data Set), CASP (Physicochemical Properties of Protein Tertiary Structure Data Set), California, blogDataTrain and the artificial datasets were 2dplanes, fried, elevators and ailerons. These datasets contain a single-target regression problem and are available at UCI repository [17]. Table 1 shows the features of the real world and artificial data sets used in the method evaluation.

Table 1. Real world datasets description

Dataset	# Examples	# Inputs
House8L	22784	8
House16H	22784	16
calHousing	20640	7
CASP	45730	9
blogDataTrain	52472	281
2dplanes	40768	10
fried	40768	10
ailerons	13750	41
elevators	8752	18

As performance measures, the mean relative error (MRE) and the mean percentage of accepted unlabeled examples (MPAUE) in the training were used. Finally, the error reduction was measured by using the relative error (in percentage) between the reference scenario (no unlabeled examples used) E_0 and the case with the parametrization that lead to the lowest error E_{lowest} (includes the reference case E_0). Equation 1 defines the Error Reduction.

$$Error\ Reduction = \frac{|E_0 - E_{lowest}|}{E_0} \tag{1}$$

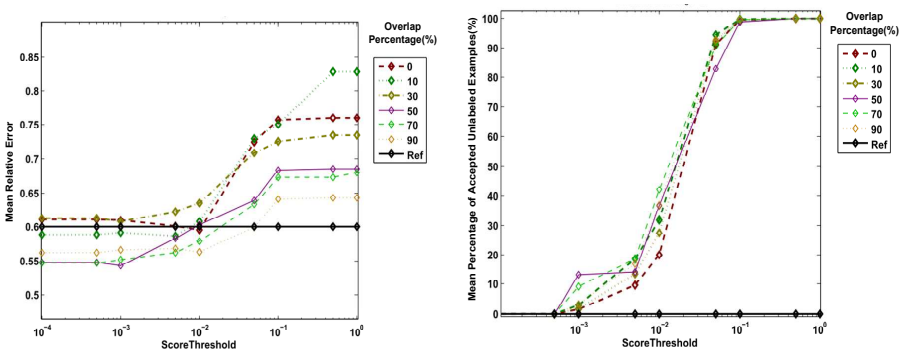
If the reference case yields the lowest error, then the Error Reduction is zero, which means that the algorithm is not useful for that particular scenario.

The algorithm was developed in the Massive Online Analysis (MOA) platform where the AMRules was developed [18]. Its an open source platform of Machine Learning and Data Mining algorithms applied to data streams. This platform was implemented in JAVA programming language.

5 Results

In this section, the evaluation results are presented and discussed. Some scenarios examples plots of MRE and MPAUE for overlap percentage and score threshold combination are presented. The reference curve (Ref) corresponds to the scenario where no unlabeled example is used in the training. Since the inputs are selected randomly, 10 runs and respective averaging of the MRE and MPAUE were performed in order to obtain more consistent values. This section also presents the error reduction for each dataset and stream unlabeled examples percentage simulation.

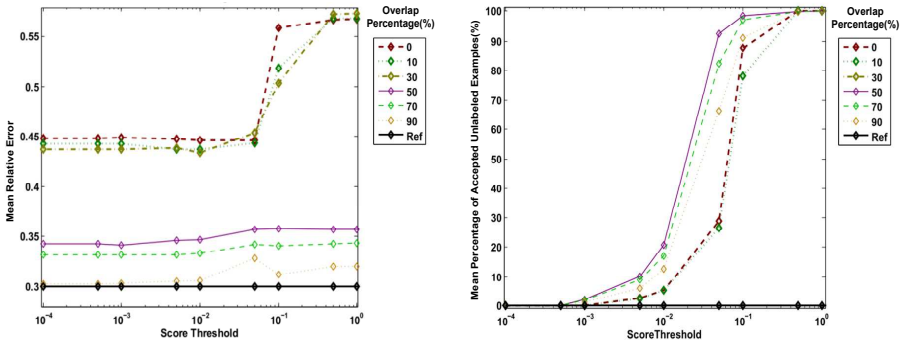
Figure 1 presents the plots of the MRE and MPAUE of a successful case for 80% of unlabeled examples stream. The curves on the left reveals that there exists some cases (combination of overlap percentage and score threshold) that lead to beneficial use of unlabeled examples. For the case of overlap of 50% and score threshold of 0.001, the use of 13.4% of the unlabeled examples in the training lead to reduction of 5,3% of the MRE in average. In general, it also observed that the overlapping decrease the MRE.



(a) (b)

Fig. 1. Mean relative error (a) and mean percentage of accepted unlabeled examples (b) for a data stream with 80% of unlabeled examples. The examples are from the House8L dataset.

Figure 2 shows a case where the algorithm does not present any combination of overlap percentage and score thresholds that lead to model improvement. In this case, most of unlabeled examples contributed to model damage and the artificial labels conveyed significant errors (all curves are above the reference curve). This fact suggests that the dataset characteristics (e.g, inputs variables distributions) may influence the performance. The error propagation through the model lead to worst predictions in the artificial labeling. This effect leads to a cycle that reinforce the error on each unlabeled example processing. In fact, the more unlabeled examples arrive the higher is the error.



(a) (b)

Fig. 2. Mean relative error (a) and mean percentage of accepted unlabeled examples (b) for a data stream with 80% of unlabeled examples. The examples are from the calHousing dataset.

Table 2 presents the error reduction of the experiments on real world datasets for each chosen unlabeled examples probabilities. When the value is zero, it means that there was not any combination of overlap percentage and score threshold that improved the model and the reference scenario is the best.

Table 2. Error reduction (%) for real world datasets.

Datasets	Unlabeled examples probabilities				
	50%	80%	90%	95%	99%
House8L	2,33	5,31	0,26	0,00	0,00
House16H	1,01	0,01	0,01	0,90	0,00
calHousing	1,11	0,00	1,62	0,01	0,00
CASP	0,8	3,45	1,06	0,00	0,00
blogDataTrain	2,54	1,56	0,26	0,00	0,00

According to Table 2, the algorithm seems to benefit most part of the scenarios. However, the benefits are in general relatively small. As expected, the more elevated the probability of unlabeled example is, the less is the relative error reduction.

Table 3 presents the error reduction for real artificial datasets in a similar way as the real world datasets presented in Table 2. The artificial datasets also present the same trend of error reduction when the probability of unlabeled example incoming increases. The error reduction is also frequently small.

In essence, the MRE curves and the error reduction tables support the view that the algorithm leads to an error reduction (despite being small) by using labeled examples in most cases. It was observed that none of the tested scenarios

Table 3. Error reduction (%) for artificial datasets.

Datasets	Unlabeled examples probabilities				
	50%	80%	90%	95%	99%
2dplanes	1,48	0,00	1,70	0,02	0,00
fried	4,64	3,21	1,04	0,70	0,00
ailerons	1,83	0,08	0,00	1,25	0,00
elevators	3,21	0,60	1,48	0,93	0,00

worked for simulated streams with 99% of unlabeled examples. In fact, this scenario is an extreme case where the model is trained essentially with artificially labeled examples and the error propagation can easily occur.

6 Conclusion

In this paper, an online semi-supervised single-target algorithm for regression based on co-training is addressed. This work prospects the development of multi-target regression algorithm that performs semi-supervised learning by co-training.

In general this co-training approach reduces the prediction error with the proper parameters calibration. The mean relative error is reduced by using a portion of unlabeled examples in most of evaluation experimental scenarios. However, the error reduction is relatively small and the parametrization depends on the dataset. It can be also conclude that, in order to obtain model improvement, only a small amount of unlabeled examples are used in the training.

As future work, this method will be extended to multi-target regression. The fact that very few unlabeled examples can lead to some improvement may suggest the study of the conditions that lead to this improvement. In order to increase the algorithm validity, the evaluation tests will be performed using a higher number of real world datasets with a significant amount of examples.

7 Acknowledgements

This work is financed under the project "NORTE-01-0145-FEDER-000020" funded by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

1. Changsheng Li, Weishan Dong, Qingshan Liu, and Xin Zhang. MORES: online incremental multiple-output regression for data streams. *CoRR*, abs/1412.5732, 2014.

2. João Duarte and João Gama. Multi-Target Regression from High-Speed Data Streams with Adaptive Model Rules. In *IEEE conference on Data Science and Advanced Analytics*, 2015.
3. Adebiyi A. Ariyo, Adewumi O. Adewumi, and Charles K. Ayo. Stock price prediction using the arima model. In *Proceedings of the 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, UKSIM '14, pages 106–112, Washington, DC, USA, 2014. IEEE Computer Society.
4. Zhi hua Zhou, Senior Member, and Ming Li. Semi-supervised regression with co-training style algorithms. *IEEE Transactions on Knowledge and Data Engineering*, page 2007.
5. Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1 - Volume 01*, WACV-MOTION '05, pages 29–36, Washington, DC, USA, 2005. IEEE Computer Society.
6. Zaid Chalabi Punam Mangtani Masahiro Hashizume Chisato Imai, Ben Armstrong. Article: Time series regression model for infectious disease and weather. *International Journal of Environment Research*, (142):319–327, June 2015.
7. Huseyin Sekerc Volkan Uslana. Article: Quantitative prediction of peptide binding afnity by using hybrid fuzzy support vector regression. *Applied Soft Computing*, (43):210–221, January 2016.
8. Pilsung Kang, Dongil Kim, and Sungzoon Cho. Semi-supervised support vector regression based on self-training with label uncertainty: An application to virtual metrology in semiconductor manufacturing. *Expert Syst. Appl.*, 51:85–106, 2016.
9. Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 908–913, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.
10. Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *CoRR*, abs/1304.5634, 2013.
11. Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.
12. Steven P. Abney. Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 360–367, 2002.
13. Sally Goldman and Yan Zhou. Enhancing Supervised Learning with Unlabeled Data. In *Proc. 17th International Conf. on Machine Learning*, pages 327–334. Morgan Kaufmann, San Francisco, CA, 2000.
14. Mohamed Farouk Abdel Hady, Friedhelm Schwenker, and Günther Palm. *Semi-supervised Learning for Regression with Co-training by Committee*, pages 121–130. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
15. Ulf Brefeld, Thomas Gärtner, Tobias Scheffer, and Stefan Wrobel. Efficient co-regularised least squares regression. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 137–144, New York, NY, USA, 2006. ACM.
16. João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.
17. K. Bache and M. Lichman. UCI machine learning repository, 2013.
18. Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, August 2010.