KeCo: Kernel-based Online Co-agreement Algorithm

Laurens van de Wiel[1a], Tom Heskes[1b], and Evgeni Levin[2]

[1] Institute for Computing and Information Sciences,
Radboud University, The Netherlands
a: firstnamelastname@student.ru.nl
b: firstname.lastname@science.ru.nl
[2] The Netherlands Organization for Applied Scientific Research,
Zeist, The Netherlands
firstname.lastname@tno.nl

Abstract. We propose a kernel-based online semi-supervised algorithm that is applicable for large scale learning tasks. In particular, we use a multi-view learning framework and a co-agreement strategy to take into account unlabelled data and to improve classification performance of the algorithm. Unlike the standard online methods our algorithm is naturally applicable to many real-world situations where data is available in multiple representations. In addition our online algorithm allows learning non-linear relations in the data via kernel functions, that are efficiently embedded into the formulation of the algorithm. We test performance of the algorithm on several large-scale LIBSVM and UCI benchmark datasets and demonstrate improved performance in comparison to standard online learning methods. Last but not least, we make a Python implementation of our algorithm available for download[3].

Keywords: kernel, non-linear, online, large-scale, semi-supervised, co-agreement, multi-view, classification

1 Introduction

Semi-supervised learning algorithms have gained more and more attention in recent years as they allow the use of large amounts of easily accessible unlabelled data. One of the most elegant approaches to take unlabelled data into account is based on *multi-view* framework [1]. Multi-view learning algorithms split the attributes into independent sets and an algorithm is learnt based on these different 'views'. The goal of the learning process consists in finding a prediction function for every view performing well on the labelled data and so that all prediction functions agree on the unlabelled data. Closely related to this approach is the *co-agreement* framework, where the same idea of agreement maximization between the predictors is central. Briefly stated, algorithms based upon this approach search for hypotheses from different views, such that the training error of

[3] Available at https://github.com/laurensvdwiel/KeCo

each hypothesis on the labelled data is small and, at the same time, the hypotheses give similar predictions for the unlabelled data. Within this framework, the disagreement among the predictors is taken into account via a co-regularization term. Empirical results show that the co-regularization approach works well for domain adaptation [2], classification [3,4], regression [5], and clustering [6] tasks. Moreover, theoretical investigations demonstrate that the co-regularization approach reduces the Rademacher complexity by an amount that depends on the 'distance' between the views [7,8].

Recently an online multi-view algorithm has been proposed in [9]. The algorithm can operate in the semi-supervised regime by using *co-regularization*. The reported performance of the algorithm is promising compared to other state-of-the-art supervised approaches that do not make use of co-regularization, such as PEGASOS [10] and SPD [11]. Our work presents substantial improvements to the online algorithm proposed in [9] by formulating a novel co-agreement learning strategy (explained in Section 3) and use of the non-linear kernel function, which enables the method to learn non-linear relations from the data.

2 Preliminaries

Prior to introducing the co-agreement framework and our algorithm we describe some standard notations and settings that are frequently used in online learning.

Consider a training set $D = (X, Y)$ of size N, originating from a set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ of data points where $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^{\mathbf{T}} \in \mathcal{X}^N$ and $Y = (y_1, \ldots, y_N)^{\mathbf{T}} \in \{-1, 1\}$. Thus, a single data point $(\mathbf{x}, y)$ where $\mathbf{x} = (x_1, \ldots, x_m)$ is a feature vector of size m, with features defined as $\mathbf{x}_i \in \mathcal{X}$ and a label $y \in \{-1, 1\}$. A *loss function* $\mathcal{L}(y, f(\mathbf{x}))$ measures the quality of the prediction $f(\mathbf{x})$ if the actual label is y. The learning task now becomes a minimization task, which constitutes a typical *binary-classification* learning setting.

In practice, we often would like not only to achieve the smallest possible loss (via error minimization) but also to ensure good generalization of the model. A popular mechanism to accomplish this is *regularization*, which is applied in state-of-the-art online learning algorithms such as GURLS [12], PEGASOS [10], SPD [11] and LPC [13].

Regularization enhances the minimization of a loss function by imposing additional restrictions on specific properties (e.g. smoothness, sparsity, etc.) of the prediction function $f(\mathbf{x})$. This is done by adding a regularization term with an appropriate parameter λ, which denotes how much regularization should be applied.

Online learning methods are frequently formulated in the primal setting for computational efficiency. However, the dual formulation (e.g. [10]) makes it possible to learn *non-linear relations* in the data. Predictions then depend on a linear combination of weights and kernel function evaluations:

$$\hat{y}_i = \frac{1}{\lambda t} \sum_{j=1, j \neq i}^{p} \alpha[j] y_j K(\mathbf{x}_i, \mathbf{x}_j) \ , \tag{1}$$

where $\hat{y}_i$ is the predicted label for example $\mathbf{x}_i$, α represents the sample weight vector, y_j is the actual label for example $\mathbf{x}_j$, λ is the regularization parameter, t is the number of observed examples at the moment of prediction, and $1/(\lambda t)$ represents the step size. Furthermore, the α-vector is a sparse vector of size N with p non-zero elements, containing discrete weight values to indicate the importance of specific examples. The kernel functions make it possible to create rich feature spaces at low computational cost.

To achieve optimal prediction in (1), [10] proposes to minimize

$$F(\alpha) = \min_{\alpha} \frac{\lambda}{2} \sum_{i,j=1}^{p} \alpha[i]\alpha[j] K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{p} \sum_{i=1}^{p} \max \left\{ 0, 1 - y_i \sum_{j=1}^{p} \alpha[j] K(\mathbf{x}_i, \mathbf{x}_j) \right\} \ , \tag{2}$$

where the last term corresponds to hinge loss. The setting above is frequently referred as minimization in a dual setting.

2.1 Multiple Views

The multi-view paradigm [1] is particularly suited for learning from datasets having more than a single data representation. A classic example is a web document classification task [1], where documents are represented via two different views - one that is based on the links and another one based on the text in each document. As another example, complex, structured data with multiple representations are frequently encountered in the biomedical domain, making multi-view methods a natural application choice. Although in many circumstances the individual data representation can be sufficient for training a model, a combination of the multiple views can lead to more robust and accurate predictions compared to the ones obtained via the individual views [14].

Online learning in the multi-view setting was recently introduced in [9]. Informally, building a prediction model for each view can be considered as training several '*judges*', whose advice can be combined for improved evaluation. Views can be built by randomly choosing various features from the dataset. However, more informative views are usually considered such as those that use different feature sources (e.g. in a medical diagnosis application, separating the blood sample data from the x-ray results).

We extend the description of an example $(\mathbf{x}, y)$ to cope with multiple views as with $\mathbf{x} = (\mathbf{x}^{(1)}, ...\mathbf{x}^{(V)})$, where V is the maximum number of views and each $\mathbf{x}^{(i)}$ represents the i^{th} subset of the features in $\mathbf{x}$. The predicted label for a view i is denoted as $\hat{y}^{(i)}$. Our learning algorithm is presented in the following section.

3 Kernel-based Online Co-agreement Algorithm

First we discuss the preliminaries on how to deal with predictions in the case of unlabelled examples.

(Semi-)supervised prediction for an example $\mathbf{x}_i$ in a single view n after t iterations of the KeCo algorithm is defined as:

$$\hat{y}_i^{(n)} = \frac{1}{\lambda(t-1)} \sum_{j=1, j\neq i}^{p} \alpha_{t-1}^{(n)}[j] z_j^{(n)} K(\mathbf{x}_i^{(n)}, \mathbf{x}_j^{(n)}) \ , \tag{3}$$

where $z_i^{(n)}$, given the example to be labelled or unlabelled, represents either the label or the co-agreement for example i:

$$z_i^{(n)} = \begin{cases} y_i & \text{if } x_i \in S \\ c_i^{(n)} & \text{if } x_i \in \hat{S} \end{cases} \ , \tag{4}$$

here S represents the set containing all labelled examples, $\hat{S}$ the set containing all unlabelled examples and $c_i^{(n)}$ is the co-agreement (as described in the next paragraph) for sample i in view n. In the case where $\hat{S} = \emptyset$, (3) turns into a supervised multi-view variant of the prediction function used in the Pegasos algorithm [10].

Co-agreement is a strategy to estimate an '*agreement*' on the label of an example $\mathbf{x}_i$. This is done by making use of different views of the examples and training on each of those views separately in order to create multiple models. We consider these models as 'judges'. Co-agreement represents the agreement between the 'judges' in order to label an example. This is done for each view n by measuring agreement of all the views over the prediction of the example, while excluding the current view n:

$$c_i^{(n)} = \text{sign}\left(\sum_{v=1, v\neq n}^{V} \hat{y}_i^{(v)} \right) \ . \tag{5}$$

We propose to extend the hinge loss with the use of the co-agreement (5) in the case of unlabelled examples as:

$$\max\{0, 1 - z_i^{(n)} \hat{y}_i^{(n)}\} \ , \tag{6}$$

with $z_i^{(n)}$ as described in (4). Again when $z_i^{(n)} \hat{y}_i^{(n)} \geq 1$, there is *zero* loss, allowing us to generalize the rule to $z_i^{(n)} \hat{y}_i^{(n)} < 1$ and retain the possibility to use it as part of the optimization goal (2).

3.1 KeCo Algorithm

We propose a kernel-based online co-agreement algorithm, which we name KeCo, that is applicable to large-scale semi-supervised binary classification tasks.

S = The set containing all labelled examples
$\hat{S}$ = The set containing all unlabelled examples
V = The number of views
λ = The regularization parameter
T = The maximum number of iterations
$\alpha_t^{(n)}$ = α-vector for view n at iteration t
$\hat{y}_i^{(n)}$ = Predicted label for example i in view n, see (3)
$\mathbf{x}_i^{(n)}$ = Feature vector for example i in view n
$z_i^{(n)}$ = Represents the label or co-agreement label for example i, see (4)

Algorithm 1 KeCo Algorithm

Input: $S, \hat{S}, V, \lambda, T$
1: **for** $t = 1, 2, \ldots, T$ **do**
2: Choose $i \in \{1, \ldots, |S \cup \hat{S}|\}$ uniformly at random
3: **for all** $j \in \{1, \ldots, |S \cup \hat{S}|\} \wedge j \neq i, n \in \{1, \ldots, V\}$ **do**
4: $\alpha_{t+1}^{(n)}[j] \leftarrow \alpha_t^{(n)}[j]$
5: **for all** $n \in \{1, \ldots, V\}$ **do**
6: $\hat{y}_i^{(n)} \leftarrow \frac{1}{\lambda t} \sum_{j=1, j \neq i}^{p} \alpha_t^{(n)}[j] z_j^{(n)} K(\mathbf{x}_i^{(n)}, \mathbf{x}_j^{(n)})$
7: **for all** $n \in \{1, \ldots, V\}$ **do**
8: **if** $z_i^{(n)} \hat{y}_i^{(n)} < 1$ **then**
9: $\alpha_{t+1}^{(n)}[i] \leftarrow \alpha_t^{(n)}[i] + 1$
10: **else**
11: $\alpha_{t+1}^{(n)}[i] \leftarrow \alpha_t^{(n)}[i]$
Output: $[\alpha_{T+1}^{(1)}, \ldots, \alpha_{T+1}^{(V)}]$

Initially, $\alpha_0^{(n)}[1] = \ldots = \alpha_0^{(n)}[\ |S \cup \hat{S}|\ ] = 0$ for all $n \in \{1, \ldots, V\}$.

In the formulation above we could consider having a regularization parameter λ for each different view. This, however, would notably increase the number of hyper parameters and hence the time required for an optimal model calculation (e.g. grid search via cross-validation).

The computational complexity of a prediction (see (3) and Algorithm 1, line 6), may require as many as $\min(t, p)$ kernel evaluations.

4 Experimental set-up and results

We evaluate the performance of KeCo and compare it with the Pegasos [10] algorithm by measuring the AUC [15] achieved on the testing data.

We have evaluated the performance during experiments on the SVMGUIDE1, SVMGUIDE3 [16] and Ionosphere datasets. SVMGUIDE1 and SVMGUIDE3 are publicly available through LIBSVM [17], Ionosphere is available through the UCI [18] repository. KeCo is designed for partially-labelled, large-scale datasets with multiple views that contain non-linear relations. These datasets do not necessarily cohere to all of these claims, but for the sake of these experiments are considered as such.

Each experiment is repeated five times and the reported AUC is an average over the performance of these five experiments. In each experiment we first randomly select 20% of the data to be considered as labelled examples and use a classic train-test split of the data, where we randomly select 70% to be used for training and the remaining 30% of the data is used for testing.

The training consists of a 10-fold cross-validation with a grid search over the parameter sets where $T = 1000$, $\lambda \in \{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and for the Gaussian kernel, $\sigma \in \{2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{3}\}$. The optimal grid found during this process is used to construct a model on the training set and that model is evaluated on the test set in order to generate the result for an experiment. We only evaluate a 2-view scheme for KeCo and we construct the views to represent a randomly selected 75% of the original set of features.

Furthermore, we used the following strategy to evaluate the performance of the algorithm. Semi-supervised strategy (KeCo) first uses 50% of the maximum iterations T to train on the labelled samples present in the partially labelled dataset, followed by a repeating sequence that considers 1 labelled followed by 1 unlabelled sample until the number of iterations reaches the value for T. The supervised strategy (Pegasos) uses the exact same sequence as KeCo, except that it skips the learning from the unlabelled sample.

The result of the experiments can be found in Table 1 where we see the Gaussian kernel outperforming the linear one and in both cases achieving a statistical significant improvement for semi-supervised learning with respect to the supervised setting. For the implementation of the experimental setup we have made use of the scikit-learn framework [19].

5 Conclusion

This work presents a kernel-based online co-agreement algorithm applicable to large scale semi-supervised classification tasks.

Our algorithm is related to online methods such as such as Pegasos [10], LaSVM [20] and GURLS [12] and unlike many of these methods is naturally applicable for multi-view datasets. In the empirical evaluation we demonstrate

Table 1. Results of the semi-supervised KeCo algorithm, in a 2-view setting, with a comparison to the fully supervised Pegasos algorithm on 20% labelled versions of the SVMGUIDE1, SVMGUIDE3 and Ionosphere datasets. The comparisons have been made using a Linear and a Gaussian kernel.

Kernel	**linear**		**Gaussian**	
Method	**Pegasos**	**KeCo**	**Pegasos**	**KeCo**
SVMGUIDE1	0.8778	0.8959	0.9231	**0.9238**
SVMGUIDE3	0.6491	0.6183	0.7192	**0.7331**
Ionosphere	0.7627	0.8012	0.8953	**0.9107**

that our method consistently performs well on publicly available datasets as well as notably outperforms supervised learning algorithms.

In particular we demonstrate that a kernel-based version of the method in combination with a co-agreement over a multi-view learning regime makes it possible to achieve more optimal results then the fully supervised state-of-the-art Pegasos algorithm, given a partially labelled dataset. Last but not least, we make available an efficient implementation of our algorithm coded in Python[4].

Our algorithm can be extended to be applicable to various learning tasks. For instance, it can straightforwardly be adapted for the task of large scale kernel-based online regression analysis.

In the near future we also aim to pursue a theoretical analysis of kernel-based online co-agreement algorithms. For example, we aim to investigate the consistency of multi-view online learning, and provide results on the rate of convergence of the algorithm as the sample size increases.

Acknowledgments. This research was financially supported by Top Institute Food and Nutrition, Wageningen, The Netherlands (Project RE-002).

References

1. Blum, A., Mitchell, T.: Combining Labeled and Unlabeled Data with Co-training. In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory. COLT' 98, New York, NY, USA, ACM (1998) 92–100
2. Kumar, A., Saha, A., Daume, H.: Co-regularization Based Semi-supervised Domain Adaptation. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., Culotta, A., eds.: Advances in Neural Information Processing Systems 23. Curran Associates, Inc. (2010) 478–486
3. Sindhwani, V., Niyogi, P., Belkin, M.: A co-regularization approach to semi-supervised learning with multiple views. In: Proceedings of ICML Workshop on Learning with Multiple Views. (2005)
4. Goldberg, A., Li, M., Zhu, X.: Online Manifold Regularization: A New Learning Setting and Empirical Study. In Daelemans, W., Goethals, B., Morik, K., eds.: Machine Learning and Knowledge Discovery in Databases. Volume 5211 of Lecture Notes in Computer Science., Springer Berlin Heidelberg (2008) 393–407

[4] Available at `https://github.com/laurensvdwiel/KeCo`

5. Brefeld, U., Gärtner, T., Scheffer, T., Wrobel, S.: Efficient Co-regularised Least Squares Regression. In: Proceedings of the 23rd International Conference on Machine Learning. ICML '06, New York, NY, USA, ACM (2006) 137–144
6. Brefeld, U., Scheffer, T.: Co-EM Support Vector Learning. In: Proceedings of the Twenty-first International Conference on Machine Learning. ICML '04, New York, NY, USA, ACM (2004) 16–
7. Rosenberg, D., Bartlett, P.L.: The Rademacher Complexity of Co-Regularized Kernel Classes. In Meila, M., Shen, X., eds.: Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics. (2007) 396–403
8. Sindhwani, V., Rosenberg, D.S.: An RKHS for Multi-view Learning and Manifold Co-regularization. In: Proceedings of the 25th International Conference on Machine Learning. ICML '08, New York, NY, USA, ACM (2008) 976–983
9. de Ruijter, T., Tsivtsivadze, E., Heskes, T.: Online Co-regularized Algorithms. In Ganascia, J.G., Lenca, P., Petit, J.M., eds.: Discovery Science. Volume 7569 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 184–193
10. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In: Proceedings of the 24th International Conference on Machine Learning. ICML '07, New York, NY, USA, ACM (2007) 807–814
11. Sculley, D.: Large scale learning to rank. In: In NIPS Workshop on Advances in Ranking. (2009) 1–6
12. Tacchetti, A., Mallapragada, P., Santoro, M., Rosasco, L.: GURLS: a Toolbox for Large Scale Multiclass Learning. In: NIPS 2011 workshop on parallel and large-scale machine learning. `http://lcsl.mit.edu/#/downloads/gurls`.
13. Fürnkranz, J., Hüllermeier, E.: Preference learning and ranking by pairwise comparison. Preference Learning **1** (2010) 65–82
14. Dasgupta, S., Littman, M.L., McAllester, D.: PAC generalization bounds for co-training. Advances in neural information processing systems **1** (2002) 375–382
15. Cortes, C., Mohri, M.: AUC optimization vs. error rate minimization. Advances in neural information processing systems **16**(16) (2004) 313–320
16. Hsu, C.W., Chang, C.C., Lin, C.J., et al.: A practical guide to support vector classification (2003)
17. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2** (2011) 27:1–27:27 Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
18. Lichman, M.: UCI machine learning repository (2013) `http://archive.ics.uci.edu/ml`.
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830
20. Bottou, L., Bordes, A., Ertekin, S.: LASVM (2009) `http://mloss.org/software/view/23/`.