Using Co-training and Self-training in Semi-supervised Multiple Classifier Systems

Luca Didaci and Fabio Roli

Dept. of Electrical and Electronic Engineering, University of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
{luca.didaci, roli}@diee.unica.it

Abstract. Multiple classifier systems have been originally proposed for supervised classification tasks, and few works have dealt with semi-supervised multiple classifiers. However, there are important pattern recognition applications, such as multi-sensor remote sensing and multi-modal biometrics, which demand semi-supervised multiple classifier systems able to exploit both labelled and unlabelled data. In this paper, the use, in multiple classifier systems, of two well known semi-supervised learning methods, namely, co-training and self-training, is investigated by experiments. Reported results on benchmarking data sets show that co-training and self-training allow exploiting unlabelled data in different types of multiple classifiers systems.

1 Introduction

The most of research on semi-supervised learning and classification focused on single classifiers, and few works have dealt with semi-supervised multiple classifier systems (MCS) [1-5]. A survey of semi-supervised learning methods for single classifiers can be found in [6]. The few works on semi-supervised MCS have proposed methods tailored to a specific MCS model which cannot be applied easily to a generic MCS, with the exception of [4] which uses a modified version of the self-training method. On the other hand, there are important pattern recognition applications, such as multi-sensor remote sensing and multi-modal biometrics, which demand semi-supervised multiple classifier systems able to exploit both labelled and unlabelled data. In this paper, the use, in MCS, of two well known semi-supervised learning methods, namely, co-training and self-training, is investigated by experiments. In particular, we extend the single-classifier versions of these algorithms to MCS, and assess the performances achievable for two widely used kinds of MCS. Section 2 first summarizes the co-training and self training methods, then two algorithms for their use in MCS are proposed. In Section 3, experiments with some benchmarking data sets are reported, and results are discussed. Section 4 draws some conclusions.

2 Co-training and Self-training in Semi-supervised Multiple Classifiers

In this section, we briefly describe two techniques for semi-supervised learning, namely, co-training an self-training, and propose their use to design semi-supervised

D.-Y. Yeung et al. (Eds.): SSPR&SPR 2006, LNCS 4109, pp. 522 – 530, 2006.

MCS. In the following, let us assume to have a set L (usually, small) of labelled data, and a set U (usually, large) of unlabelled data.

2.1 Co-training and Self-training

A co-training approach to semi-supervised learning and classification was proposed by Blum and Mitchell in 1998 [7]. Co-training was proposed for two classifiers and assumes that input features are naturally subdivided into two sets, and each feature subset is sufficient to train an optimal classifier, supposed that enough labelled data are available. Two separate classifiers, one for each feature subset, are trained on the initial, small, labelled data set L. It is assumed that the classifiers will exhibit a low, but better than random, accuracy. Each classifier is then applied to the unlabelled examples in U. For each classifier, the unlabelled examples classified with the highest confidence are added to the labelled data set L, so that the two classifiers can contribute to increase the data set L. Both classifiers are re-trained on this augmented data set, and the process is repeated a given number of times. The rationale behind co-training is that a classifier may assign correct labels to certain examples while it may be difficult for the other classifier to do so. Therefore, each classifier can increase the training set with examples which are very informative for the other classifier.
It is worth pointing out two fundamental assumptions of the co-training method:

1) patterns must be represented with two distinct "views", namely, with two distinct feature sets, and either feature subsets must be sufficient to design an optimal classifier if we have enough labelled data. We need the feature subsets to be conditionally independent so that the examples which are classified with high confidence by one of the two classifiers are *i.i.d.* samples for the other classifier.
2) the classifiers must be "compatible". Compatibility implies that, if we have enough labelled data in the training set, the classifiers C_1 and C_2 provide the same classification labels for all the possible test patterns. A relaxed form of this hypothesis ("partial compatibility") can be also accepted.

In self-training a classifier is initially trained using the labelled data set L. This classifier is then used to assign pseudo-class labels to a subset of the unlabelled examples in U, and such pseudo-labelled data are added to L. Usually, the unlabelled data classified with the highest confidence are selected to increase L. Then the classifier is re-trained using the increased data set L. As the convergence of this simple algorithm can not be guaranteed in general, the last two steps are usually repeated for a given number of times or until some heuristic convergence criterion is satisfied.

2.2 Semi-supervised Multiple Classifiers Using Co-training and Self-training

Co-training of Multiple Classifiers. As pointed out above, the co-training method was introduced under the assumptions of patterns represented with two distinct "views" and "compatible" classifiers. Therefore, co-training can be naturally applied to classifier ensembles made up of compatible classifiers which have distinct feature sets as input. However, such assumptions are not satisfied in many practical cases. On

the other hand, Goldman and Zhou showed that co-training also works with two classifiers using the same features [5]. Here we propose to investigate the use of co-training for a generic MCS whose classifiers can be created with different methods (e.g., using different classification algorithms or the bootstrap technique). Figure 1 shows the main step of our extended co-training algorithm applied to multiple classifiers. It should be noted that a similar algorithm is described in [11].

Co-training of Multiple Classifiers

- Given:
  - a set L of labelled training examples
  - a set U of unlabelled examples
  - the P_j prior probabilities of the underlying class distributions
- Create a set U' of examples by choosing *u* examples at random from U
- Loop for *k* iterations:
  - Build N classifiers C_i, i=1…N, using L
  - Run C_1, … C_N on U'
  - For each classifier C_i, for each class j,
    select the $n_j \propto P_j$ examples assigned to the class j by the classifier C_i with the highest classification confidence.
  - Add these $N \times \sum_j n_j$ self-labelled examples to L
  - Randomly choose $N \times \sum_j n_j$ examples from U to replenish U'
- Return a MCS using $\{C_i\}$ as base classifiers

Fig. 1. The extended co-training algorithm applied to multiple classifier systems

Given a set L (usually, small) of labelled data, and a set U (usually, large) of unlabelled data, a set U' is created by choosing *u* examples at random from U. The following steps are executed for a fixed number of iterations. N classifiers are trained on the initial, small, labelled data set L. In order to create different classifiers several methods are possible. For example, different classification algorithms or the bootstrap technique can be used. Each classifier is then applied to the *u* unlabelled examples in U'. For each classifier C_i and for each class j, n_j examples assigned to the class j by the classifier C_i with the highest classification confidence are selected and they are added to the labelled data set L. The number n_j of selected examples assigned to the class j is proportional to the prior probability of the class j. Accordingly, if there are no classification errors in the selected examples, the set of the selected examples has the same prior probabilities of the underlying distributions, and the prior probabilities of L remain unchanged. Each classifier selects $\sum_j n_j$ patterns, so that $n = N \times \sum_j n_j$ patterns are moved from U' to L. An equal number n of patterns are randomly chosen from U to replenish U'. For the next iterations, all the classifiers are re-trained on the augmented data set L, and the process is repeated a given number of times. At the end of this iterative process, an MCS using $\{C_i\}$ as base classifiers is created. In this work only MCS based on fixed rules [8] (mean, product and majority voting rule) are used.

Ensemble-Driven Self-training. In order to extend the use of the self-training method to MCS, we propose to use the concept of "ensemble-driven" self-training. Each classifier is not self-trained, but it is trained with the examples which are labelled by the MCS. In other words, the MCS is used to assign pseudo-class labels to a subset of the unlabelled examples in U, and such pseudo-labelled data are added to L. Then each classifier of the ensemble is re-trained using the increased data set L. It is worth noting that the self-supervised classifier ensemble proposed by El Gayar exploits the same concept [4], and also the extension of co-training, called "democratic" co-training, proposed by Zhou and Goldman can be regarded as a type of ensemble-driven self-training [5]. Figure 2 shows the main step of our "ensemble-driven" self-training algorithm applied to multiple classifiers.

Ensemble-driven Self -training of Multiple Classifiers

• Given:
- a set L of labelled training examples
- a set U of unlabelled examples
- the P_j prior probabilities of the underlying class distributions

• Create a set U' of examples by choosing u examples at random from U
• Loop for k iterations:
- Build N classifiers C_i, i=1…N, using L
- Create a MCS using $\{C_i\}$ as base classifiers
- Run the MCS on U'
- For each class j,
  select the $N \times n_j$ examples assigned to the class j
  by the MCS with the highest classification confidence ; $n_j \propto P_j$
- Add these $N \times \sum_j n_j$ self-labelled examples to L
- Randomly choose $N \times \sum_j n_j$ examples from U to replenish U'

• Return the MCS

Fig. 2. The extended, ensemble-driven, self-training algorithm applied to multiple classifier systems

The MCS is applied to the u unlabelled examples in U'. For each class j, $N \times n_j$ examples assigned to the class j by the MCS with the highest classification confidence are selected and they are added to the labelled data set L. As in the co-training algorithm, the number $N \times n_j$ of selected examples assigned to the class j is proportional to the prior probability of the class j, in order to do not change the prior probabilities of L. For each step $n = N \times \sum_j n_j$ patterns are moved from U' to L. An equal number n of patterns are randomly chosen from U to replenish U'. For the next iterations, all the classifiers are re-trained on the augmented data set L, and the process is repeated a given number of times. At the end of the iterative process the

'self-trained' MCS is returned. It is worth noting that in the co-training algorithm each of the N classifiers selected and labelled n_j patterns per class, so that the number of patterns labelled at each step was $n = N \times \sum_j n_j$. As in the ensemble-driven self-training algorithm patterns are selected and labelled directly by the MCS, $N \times n_j$ pattern per class are selected for each step of the algorithm.

3 Experimental Results

3.1 Data Sets and Experimental Protocol

The performances of the algorithms described in Figures 1 and 2 clearly depend on the classifier ensemble used. The goal of our experiments was to assess such performances using two widely used methods for creating a MCS, and with different data sets. In particular, we assessed performances with classifier ensembles generated by the bootstrap method and using different type of classification algorithms. Table 1 describes the main characteristics of the data sets used, and reports the size of L expressed as percentage of the available training set and as number of patterns. The datasets Letter, BCW and Optdigits come from the UCI Machine Learning repository (http://www.ics.uci.edu/~mlearn/MLRepository.html).

Table 1. Main characteristics of the data sets in terms of the number of classes, features, patterns, and size of the training set L as percentage (%L) of the available training set (the number of patterns in L is given in brackets)

Data set	Classes	Features	Patterns	%L	Reference
Gaussian	2	15	1000	5% (35)	Gaussian data set proposed in [9]
Letter	26	16	20000	15% (2100)	Letter Image Recognition
BCW	2	9	683	5% (23)	Wisconsin Breast Cancer
Optdigits	2	15	3823	5% (190)	Optical Recognition of Handwritten Digits
Feltwell	5	15	10944	2% (100)	Feltwell [10]

Each data set was subdivided randomly into a training set (30% of the patterns) and a test set (70% of the patterns). For the Feltwell data set we maintained the original subdivision in training and test sets, because a random subdivision is known to create an artificial, almost trivial, classification task [10]. Each training set was subdivided randomly into a set L (the labelled data set) and a set U (the unlabelled data set). Each experiment was repeated 10 times, with different random choices of the labelled data set L. For each data set, Table 2 shows the classifier ensembles used. In the case of ensembles generated by bootstrap Table 2 gives the base classifier used. Five different bootstrap replicas of L, that is, ensembles made up of five classifiers, were used. The terms "linearG" and "quadratic" indicate the linear and quadratic Gaussian classifiers. The term "linearLog" indicates the linear classifier that maximize the likelihood criterion using the logistic function. For each step of the algorithms in Figures 1 and 2,

$n = N \times \sum_j n_j$ patterns are selected. The value of n_j is $n_j = \alpha \cdot P_j$ where P_j is the prior probability of the class *j* and α is an integer value between 3 and 6. The size *u* of the set U' was chosen as $u = \beta \times n$ where β is 4 or 5. In other words, the set U' is β times greater than the number of patterns selected from U'. The number of iterations *k* is chosen as following: for large datasets (Feltwell, Letter) *k* is chosen as 2/3 of the maximum of the possible iterations, that is, as 2/3 of the number of iterations that emptied the set U of unlabelled patterns. For small datasets, the algorithm goes on until there are no more patterns in U.

Table 2. Classifier ensembles and base classifiers used for each data set

Dataset	*Classifier ensemble*	*Base classifier used for bootstrap*
Gaussian	1)[perceptron, linearG, quadratic] 2)[perceptron, perceptron, quadratic]	linearG
Letter	[*k*-nn (k=1); *k*-nn(k=5); parzen]	*k*-nn
BCW	[linearG, linearLog, parzen]	linearG; Parzen
Optdigits	[*k*-nn; MLP; parzen]	*k*-nn; MLP; Parzen
Feltwell	[*k*-nn, MLP, quadratic]	Quadratic; *k*-nn

For each data set, the classifiers were chosen so that the classification error obtained using base classifiers trained on L data set was substantially higher than the error obtained using base classifiers trained on the full training set $L \cup U$. In other words, we selected classifiers for which we expect to obtain a decrease of the error if the semi-supervised mechanism correctly labelled all the patterns of U. For the experiments with co-training, in order to fulfil the compatibility hypothesis, we choose classifiers that agree in their decisions at least for the 90% of the pattern of U when they are trained on the full training set $L \cup U$.

3.2 Results

Tables 3 and 4 report the results of the experiments with the algorithms of Figures 1 and 2. In particular, Table 3 refers to the experiments with classifier ensembles generated by bootstrap, while Table 4 reports the results obtained with ensembles made up of different classifiers (second column of Table 2). The reported values are the test-set error values averaged over ten runs, with different random choices of the initial labelled data set L. Results obtained with different rules for classifier combination are reported.

The 'start' column reports the percentage value of the error before the exploitation of the unlabelled data, that is, using classifiers trained only on the labelled data set L. The Δ column represents the variation of the error dues to the exploitation of the unlabelled data. Positive values of Δ indicate a reduction of the error. As the trend of the error can be non monotonic, we report the value of Δ after a fixed number of iterations of the semi-supervised process (Δ(end)) and the maxim reduction of the error during the semi-supervised process (Δ(best)). Results where Δ(best) >> Δ(end),

that is, results for which after a first reduction of the error the error increased, are reported in bold. This non-monotonic trend of the error was observed for the Feltwell dataset (Table 3), and for Gaussian dataset (Table 4). In particular, for the Feltwell dataset, after a first reduction of the error, the co-training and self-training algorithms sometimes provided an error greater than the initial test error (negative values of Δ).

Table 3. Experiments with classifier ensembles generated by bootstrap

		Co-training			**Self-training**		
		start	**Δ (best)**	**Δ(end)**	start	**Δ(best)**	**Δ(end)**
Mean rule	bcw, linearG	12.21	7.65	7.35	11.27	8.24	7.65
	bcw, parzen	11.76	7.30	7.06	5.59	3.09	2.65
	feltwell, quadratic	18.81	**4.43**	**-3.37**	18.86	**4.05**	**1.83**
	feltwell, *k*-nn	18.83	**3.28**	**-1.79**	16.54	2.39	1.10
	letter	14.33	1.49	1.31	15.84	1.77	1.58
	optdigits, *k*-nn	7.05	3.43	3.29	7.14	3.77	3.76
	optdigits, MLP	10.53	4.02	3.92	9.71	3.27	3.27
	optdigit, parzen	7.92	3.66	3.39	8.68	5.12	5.02
	gaussian	18.83	7.67	7.07	18.83	7.10	6.47
Majority voting rule	bcw, linearG	14.51	9.85	9.66	13.38	10.00	9.26
	bcw, parzen	14.17	9.80	9.51	4.90	2.45	1.67
	feltwell, quadratic	18.98	**4.67**	**-3.25**	16.63	**2.53**	**-4.63**
	feltwell, *k*-nn	19.50	**3.34**	**-1.50**	18.30	3.32	2.72
	letter	14.12	0.97	0.68	15.97	1.56	1.31
	optdigits, *k*-nn	7.32	3.38	3.16	8.21	4.99	4.80
	optdigits, MLP	11.27	4.58	4.40	10.61	4.23	4.10
	optdigit, parzen	8.16	3.90	3.64	6.86	2.82	2.31
	gaussian	18.88	7.62	7.08	19.80	9.37	9.00
Product rule	bcw, linearG	12.25	7.75	7.40	9.02	5.78	5.39
	bcw, parzen	12.99	8.53	8.28	3.04	0.69	0.05
	feltwell, quadratic	20.69	**6.37**	**-1.53**	19.70	6.01	5.12
	feltwell, *k*-nn	18.82	**3.31**	**-1.79**	14.07	**1.34**	**-0.17**
	letter	14.36	1.50	1.34	16.59	1.30	1.02
	optdigits, *k*-nn	7.05	3.43	3.29	6.94	3.90	3.90
	optdigits, MLP	13.77	7.18	7.11	10.18	4.16	4.02
	optdigit, parzen	7.15	3.33	3.12	7.67	2.50	2.48
	gaussian	19.55	8.42	7.78	19.67	8.27	7.33

In order to show the typical trend of the test-set error during the semi-supervised process, in Figure 3 we report the error on Letter and Optdigits data sets as a function of the number of iterations of the self-training algorithm applied to classifier ensembles generated using different base classifiers. Similar trends were obtained with the co-training algorithm and for the other data sets.

Table 4. Experiments with ensembles made up of different classifiers (second column of Table 2)

		Co-training			**Self-training**		
		start	**Δ (best)**	**Δ(end)**	start	**Δ(best)**	**Δ(end)**
Mean rule	bcw	7.94	4.36	3.97	6.13	3.63	3.14
	feltwell	15.63	2.99	2.26	13.01	2.12	1.42
	letter	14.24	1.47	1.33	14.39	1.21	1.06
	optdigits	7.48	3.52	3.49	7.80	4.62	4.60
	Gaussian 1)	15.83	**5.30**	**2.93**	17.53	**8.57**	**6.60**
	Gaussian 2)	18.83	**6.00**	**4.50**	18.93	**4.43**	**2.00**
Majority voting rule	bcw	8.09	4.56	4.22	7.45	4.31	4.02
	feltwell	14.99	3.59	2.46	13.05	2.83	2.46
	letter	14.29	1.54	1.44	14.30	0.95	0.87
	optdigits	7.51	3.56	3.53	7.63	4.18	4.05
	Gaussian 1)	15.80	**5.33**	**3.33**	16.70	6.57	6.10
	Gaussian 2)	18.87	**5.73**	**3.13**	17.93	2.33	0.30
Product rule	bcw	6.96	3.04	2.30	6.37	3.24	2.55
	feltwell	17.57	4.96	4.38	16.34	4.04	2.64
	letter	14.27	1.51	1.39	14.09	1.46	1.41
	optdigits	7.47	3.54	3.48	7.27	3.85	3.75
	Gaussian 1)	27.73	**16.47**	**14.77**	33.17	**18.23**	**16.57**
	Gaussian 2)	31.80	18.60	17.67	23.90	7.03	5.93

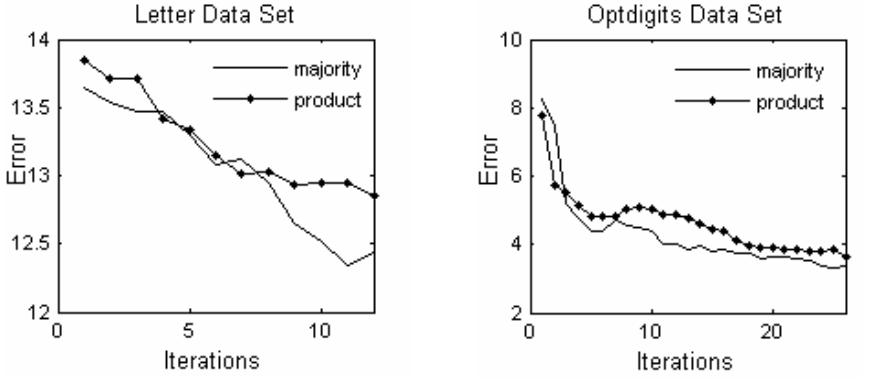

Fig. 3. Examples of the test-set percentage error as function of the number of iterations of the self-training algorithm applied to classifier ensembles generated using different base classifiers

4 Conclusions

This paper's goal was to investigate by experiments the use, in MCS, of two well known semi-supervised learning methods, namely, co-training and self-training. Although final conclusions cannot be drawn on the basis of the limited set of reported

experiments, we believe that this work made a first step toward the systematic use of co-training and self-training to design semi-supervised MCS. Reported results show that the extended versions of the co-training and self-training we proposed allow exploiting unlabelled data in two different types of multiple classifiers systems. In addition, our results confirmed a claim of other researchers [5], that is, co-training algorithm can be used even if different feature subsets are not available for the task at hand. As a future work we will continue our experimental investigation and will investigate the trade-off between the "complementarity" of classifiers, useful for MCS, and the request of "compatibility" of the co-training algorithm.

References

1. Roli F.: Semi-supervised Multiple Classifier Systems: Background and Research Directions. 6th Int. Workshop on Multiple Classifier Systems (MCS 2005), Seaside, CA, USA, June 13-15 2005, N.C. Oza, R. Polikar, J. Kittler, F. Roli Eds., Springer-Verlag, LNCS 3541, pp. 1-11
2. D'Alchè-Buc F., Grandvalet Y., Ambroise C.: Semi-supervised marginboost, Neural Information Processing Systems Foundation, NIPS 2002, 2002.
3. Bennet K., Demiriz A., Maclin R.: Exploiting unlabeled data in ensemble methods, Proc. 8th ACM SIGKDD Int. Conf. On Knowledge Discovery and Data Mining, 2002, pp. 289-296
4. N. El Gayar.: An Experimental Study of a Self-Supervised Classifier Ensemble, International Journal of Information Technology, Vol. 1, No. 1, 2004.
5. Y Zhou, S Goldman.: Democratic Co-Learning, Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), pp 594-602
6. X. Zhu, Semi-supervised learning literature survey, Technical report, Computer Sciences TR 1530, Univ. Wisconsis, Madison, USA, Jan. 2006.
7. Blum A., Mitchell T.: Combining labeled and unlabeled data with co-training, Proc. of the Workshop on Computational Learning Theory, 1998, pp. 92-100.
8. Kittler J., Hatef M., Duin R. P. W. and Matas J.: On Combining Classifiers , IEEE Trans. on Patt. Anal. and Machine Intell., Vol. 20, No. 3, pp. 226-239, March 1998
9. Skurichina M. and Duin R. P. W.: Bagging, Boosting and the Random Subspace Method for Linear Classifiers. Pattern Analysis & Applications (2002)5:121–135
10. Giacinto G., Roli F., Bruzzone L.: Combination of neural and statistical algorithms for supervised classification of remote-sensing images, Pattern Recongition Letters, Vol. 21, No. 5, 2000, pp. 385-397.
11. Solyman M., El Gayar N.F.: A Co-training Approach for Semi-supervised Multiple Classifiers, INFO 2006, 4th Int. Conference, Cairo, Egypt, 25-27 March 2006, in press.