On–Line Laplacian One–Class Support Vector Machines

Salvatore Frandina, Marco Lippi, Marco Maggini, and Stefano Melacci

Department of Information Engineering and Mathematical Sciences,
University of Siena, Italy
{frandina,lippi,maggini,mela}@diism.unisi.it

Abstract. We propose a manifold regularization algorithm designed to work in an *on–line* scenario where data arrive continuously over time and it is not feasible to completely store the data stream for training the classifier in *batch* mode. The On–line Laplacian One–Class SVM (OLapOCSVM) algorithm exploits both positively labeled and totally unlabeled examples, updating the classifier hypothesis as new data becomes available. The learning procedure is based on conjugate gradient descent in the primal formulation of the SVM. The on–line algorithm uses an efficient buffering technique to deal with the continuous incoming data. In particular, we define a buffering policy that is based on the current estimate of the support of the input data distribution. The experimental results on real–world data show that OLapOCSVM compares favorably with the corresponding batch algorithms, while making it possible to be applied in generic on–line scenarios with limited memory requirements.

Keywords: On–line learning, One–Class SVM, RKHS, Manifold Regularization, Semi–supervised learning.

1 Introduction

Nowadays, many applications, ranging from computer vision (e.g. a camera mounted on a moving robot) to sensor networks (e.g. the measurement data collected by a sensor network), have access to a huge amount of data. In particular, many scenarios are typically *on–line*, where the data arrive continuously in time and an intelligent decision system must analyze them and take its consequent actions in synchrony with the input flow. In these cases, the capabilities of the standard *batch* machine learning algorithms are limited by the available resources both in terms of time and memory. Much attention has been paid to on–line machine learning algorithms that are able to work in an incremental and never ending learning scenario [8]. As new data becomes available, the current learned hypothesis must be continuously improved by efficiently exploiting the data seen up to the current time step and the training process must be able to adapt to eventual drifts in the data distribution. Moreover, most of the real–world problems are naturally semi–supervised, since there is only a limited quantity of labeled data and an abundant quantity of unlabeled points

V. Mladenov et al. (Eds.): ICANN 2013, LNCS 8131, pp. 186–193, 2013.

[2]. Goldberg et al. [5] have proposed an on–line approach to Laplacian SVMs, extending the original idea of on–line SVMs [8] to the semi–supervised setting. Different strategies are proposed to control the time and memory requirements of the algorithm.

In many real-world problems, there is an intrinsic asymmetry in the data labeling process. In general, only a few examples can be clearly and easily labeled as belonging to a given class, whereas it may be costly to provide a label for all the available data or to specify all the relationships of each example and each class. For instance, in anomaly detection tasks the evident and detected anomalies can be tagged, while most of the events will not have a clear classification. Similarly, in multi–class classification both the number of classes and their relationships (eventual intersections, inclusions and exclusions) may not be explicitly defined. Hence, we consider a setting where a classifier has to be learned only from the positive examples that describe the class to be modeled. In the context of kernel machines, this problem has been faced by One–Class SVM (OCSVM), that is essentially a density estimation algorithm sometimes referred to as *novelty detection* algorithm [14]. In this scenario, OCSVM tries to estimate the class distribution only from the available positive examples [11]. Two straightforward on–line extensions of the density estimation algorithm are proposed in [8,6].

This paper proposes an approach referred to as *On–line Laplacian One–Class SVM* (OLapOCSVM), that extends the Laplacian SVM (LapSVM) algorithm [9] to the one–class setting considering an on–line framework. In details, the continuous stream of data is efficiently bufferized by means of a pruning strategy that decides if a new example has to be added to the buffer depending on the current estimate of the support of the input data distribution. Such strategy allows us to guarantee the convergence of the learning algorithm and to exploit standard convex programming techniques, instead of a stochastic adjustment of the parameters with exponentially decaying learning rates [3,5], that may be harder to tune. In particular, we focus on the conjugate gradient descent in the primal formulation of the SVM problem [9]. The proposed buffering technique is quite general and can be applied to any semi–supervised learning algorithm.

The paper is organized as follows. The next section introduces the learning algorithms and the buffer management techniques. Then, section 3 reports the experimental comparison of the proposed algorithm with the batch approach. Finally, section 4 draws the conclusions and delineates the future developments.

2 On–Line Laplacian One–Class SVM

The considered learning scenario consists in a stream of data for which supervisions are provided only for some examples belonging to a class of interest. Hence the goal of the training algorithm is to develop the class model on–line while the data is made available, taking advantage of both the supervised and the unsupervised samples. The first aspect to be taken into account is that only positive examples are provided by the supervisions. OCSVM learning has been devised to approach this task when a batch of positive examples is available.

The learning goal is defined by the minimization of the following functional with respect to the function $f : \mathcal{X} \to I\!R$ in a given RKHS $\mathcal{H}$ and the bias $b \in I\!R$,

$$E_{\text{oc}}[f, b] = \lambda_r \|f\|_{\mathcal{H}} + \frac{1}{|\mathcal{L}^+|} \sum_{\mathbf{x}^s \in \mathcal{L}^+} \max(0, 1 - f(\mathbf{x}^s) - b) + b \ , \tag{1}$$

where $\mathcal{L}^+$ is the set of supervised positive examples, and $\lambda_r > 0$ weights the regularization contribution. Once the function f is estimated, an input example $\mathbf{x} \in \mathcal{X}$ is assigned to the modeled class if $f(\mathbf{x}) + b \geq 1 - \xi$, where $\xi \in [0, 1]$ is a tolerance parameter[1].

In an on–line setting the data are incrementally made available in time and the classifier should be able to provide its predictions at each time step t. Hence, the learning algorithm must be designed to yield an optimal solution at each t, given the data seen up to that instant. However, since the horizon may potentially be infinite, it is unfeasible to collect all the data up to t and to train the classifier on them. In fact, this approach would require an unbounded amount of memory and increasingly longer training times for the classifier. Moreover, the memorization of all the data history may degrade the performances in those cases in which there is a drift in the data distribution. A classical approach to on–line learning is to exploit a finite size buffer that collects only part of the incoming data [8,5]. This technique requires to define an appropriate policy to manage the buffer overflow. Depending on the specific task, when a new example has to be added to the buffer different criteria can be taken into account, especially when the buffer is full and the new example should eventually replace a memorized one. In the case of replacement, a simple criterion is to remove the oldest stored example, thus implementing a sliding window on the input stream. A better solution is to consider also the significance of the examples for training the classifier [5]. More advanced methods try to estimate the impact of each buffer replacement operation on the accuracy of the estimated classifier [12].

The proposed method is based on a pruning strategy that decides if the new example is to be added to the buffer depending on the current estimate of the support of the input data distribution. Given a tolerance $\epsilon > 0$, the example at time t, $\mathbf{x}_t$, is added to the buffer $\mathcal{B}_t$ only if the condition

$$\forall(\mathbf{x}_b \in \mathcal{B}_{t-1}) : \ \|\mathbf{x}_t - \mathbf{x}_b\|_p > \epsilon \tag{2}$$

is satisfied, otherwise $\mathcal{B}_t = \mathcal{B}_{t-1}$. The norm used to compute the data similarity can be chosen given the specific characteristics of the input space $\mathcal{X}$ (e.g. for high–dimensional spaces $p \leq 1$ may be considered instead of the Euclidean norm with $p = 2$ [1]). This solution allows us to provide a stable coverage of the support of the input data distribution with a given resolution depending on the parameter ϵ. Clearly, a better approximation of the support is obtained for smaller values of ϵ, at the cost of a larger amount of needed memory space.

[1] This is a slightly modified formulation of the original OCSVM. The offset of 1 in the hinge loss is added to enforce a larger value of f when evaluated on the training examples and then the decision function is relaxed by ξ.

The pruning technique can be combined with replacement criteria as those listed before, when a maximum buffer size is set. In the experimental settings, we employed a buffer whose insertions are managed by the pruning technique defined by the rule of eq. (2) without limitations on its maximum size. Notice that the proposed technique allows us to filter out also potential replicates of already processed data points.

Finally, the unsupervised examples from the input stream can be exploited to provide additional information on the data distribution. By following the Laplacian SVM approach [2,9] we can add a manifold regularization term to the objective function. First, given a set of unsupervised points $\mathbf{x}^u \in \mathcal{U}$, we build the Laplacian graph $L = D - W$ associated to the set $\mathcal{M} = \mathcal{L}^+ \cup \mathcal{U}$. W is the data adjacency matrix, whose entry w_{ij} measures the *similarity* of the examples $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{M}$, and D is the diagonal matrix with the degree of each node, i.e. $d_{ii} = \sum_{j=1}^{|\mathcal{M}|} w_{ij}$. The manifold regularization term is defined as

$$E_{\mathrm{g}}[f] = \|f\|^2_{\mathcal{M}} = \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ i \neq j}} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2 . \tag{3}$$

In general, several choices of $\|f\|^2_{\mathcal{M}}$ are possible and, in the considered case, $\|f\|^2_{\mathcal{M}} = f^T L f$, as it can be easily derived from eq. (3) [2].

Since we are considering a one–class approach, the manifold construction is modified accordingly. In particular, the graph Laplacian must contain only the connections among the positive examples and the unsupervised ones that are most similar to them. This choice is motivated by the fact that the goal of the one–class SVM algorithm is to model the support of the distribution only for the positive samples and the presence of out–of–class–distribution data may hinder the class modeling. The unsupervised examples that have no connections in the Laplacian graph will be neglected in the learning process. We avoid using the k-NN rule to build W (as frequently done in Laplacian SVMs [2]) since it would result in connecting at least k neighbors of each point $\mathbf{x}_i$ of $\mathcal{M}$, even if some of them are far away from $\mathbf{x}_i$. Moreover the k–NN rule is known to suffer from several problems in high dimensional spaces [13]. In this work, we add a connection between two examples $\mathbf{x}_j$ and $\mathbf{x}_i$ only if $\|\mathbf{x}_i - \mathbf{x}_j\|_p < \epsilon_L$, for a given tolerance ϵ_L. The connection weight w_{ij} is then computed using an exponential decay $w_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_p / (2\sigma_L))$, where σ_L is a hyper–parameter.

This choice also allows us to efficiently handle the on–line incremental building of the adjacency matrix W. As a matter of fact, in the on–line setting W is computed on the points of $\mathcal{B}_t$, and it must be updated every time that $\mathcal{B}_t$ differs from $\mathcal{B}_{t-1}$. When a new point $\mathbf{x}$ is added to the buffer, we have already computed its distance from the other points in $\mathcal{B}_t$, due to the proposed buffering strategy (2). Since we use the same distance $\|\cdot\|_p$ both to manage the buffer and to build w_{ij}, no additional distance computations are needed to update the matrix W. Using the k-NN rule would require to store information on the k nearest neighbors of each point in $\mathcal{M}$ to update W.

Considering both the criteria of eq. (1) and eq. (3), the learning objective is to minimize the following functional with respect to f and b

$$E[f, b] = E_{\text{oc}}[f, b] + \lambda_m E_{\text{g}}[f] \ , \tag{4}$$

where $\lambda_m \geq 0$ weights the Laplacian contribution. By applying the Representer Theorem, the solution at the time step t can be written as $f(\mathbf{x}) = \sum_{\mathbf{x}_j \in \mathcal{B}_t} \alpha_j^t K(\mathbf{x}_j, \mathbf{x})$, where $K(\cdot, \cdot)$ is the kernel of the considered RKHS [9]. The optimal parameters α_j^t are determined by applying a conjugate gradient descent technique given the convex objective function of eq. (4). Actually, the computation is needed only when a change is applied to the buffer (insertion or replacement) and the starting point for the gradient descent is chosen such that $\alpha_j^t = \alpha_j^{t-1}$ for those examples such that $\mathbf{x}_j \in \mathcal{B}_t \cap \mathcal{B}_{t-1}$. Due to this "warm start" of the optimization procedure between consecutive time instants, only a few iterations are needed to converge to the minimum of eq. (4).

3 Experimental Results

The evaluation was performed on the MNIST and NSL–KDD datasets[2] . MNIST is a widely–used digit classification benchmark composed of 70,000 points, and we considered the classification setting used in [5], based on two binary classification problems (digit 0 vs. 1 and 1 vs. 2). The evaluation follows the one–class approach, and, hence, the classifier is trained only on the positive class, that is, digit 0 for 0 vs. 1, and digit 1 for 1 vs. 2. In the experiments, the original training and test sets were used without any preprocessing, except a normalization of the gray–level value of each pixel. NSL–KDD is an anomaly detection dataset, derived from the KDD Cup 1999 benchmark by removing several problems from the original data [15]. This dataset consists of $\approx$ 95,000 examples describing network traffic at different timestamps, each one labelled with one of following classes: DoS, R2L, U2R, probing and normal (no attack). In the experiments, the R2L class was neglected, as it contains too few examples (0.07% for the training set and 0.37% for the test set). Each continuous attribute was discretized into 10 bins, by selecting the quantization thresholds using a maximum entropy principle (each bin should contain roughly the same number of examples). Both the datasets come divided into a training and a test split. The test split was used to evaluate the classifier quality, whereas we divided the training split into two portions. The first one (10% of the data) was used in an off–line setting to cross-validate the hyper–parameters of the classifiers[3]. The second portion (90% of the data) was streamed in an on–line fashion and processed by the proposed algorithm. In both cases, only 10% of the points were supervised. A Gaussian kernel was selected and its width σ_K was chosen in a discrete grid defined in $[3, 21]$ with step size 3; the regularization parameter λ_r was selected in $\{10^{-h} | h = 1, \ldots, 6\}$;

[2] `http://yann.lecun.com/exdb/mnist/`; `http://iscx.ca/NSL-KDD/`

[3] Two thirds of the validation examples were exploited to train the classifier, whereas one third of the validation examples were used to evaluate the classifier accuracy.

the Laplacian regularization parameter λ_m was chosen in $\{0, 10^{-h} | h = 0, \ldots, 6\}$; the width of the Gaussian for computing the Laplacian graph weights σ_L and the distance tolerance ϵ_L were chosen in $[3, 12]$ with a step of 3 for the MNIST dataset, and in $\{0.5, 1, 1.5, 2, 3\}$ for the NSL–KDD dataset.

The first experiment was aimed at evaluating different settings of the pruning policy used for the buffer management. In particular, three different norms were considered (i.e. $p \in \{0.5, 1, 2\}$) and the resulting size of the buffer was computed by varying the resolution ϵ used to approximate the data distribution support. Table 1 reports the total number of elements N of the streamed training set that are stored into the buffer for the different norms and tolerance values. In particular, the table shows that larger values of ϵ are needed for the L_1 and $L_{0.5}$ norms to yield the same number of stored examples with respect to the L_2 norm. This behavior confirms the property that, when the norm index p decreases, the average distances among the points tend to increase [1]. By an appropriate choice of the combination of the norm and the related tolerance ϵ, three different settings were defined for the following experiments. In particular, in *Online setting 1* the tolerance is chosen such that roughly the 15% of the streamed examples are stored in the buffer for training the classifier; in *Online setting 2* the buffer contains about the 10% of the data, whereas in *Online setting 3* only the 5% of the available examples is memorized into the buffer.

Table 1. Number of examples N stored into the buffer with respect to the norm (L_2,L_1 and $L_{0.5}$) and the tolerance parameter ϵ, for the MNIST and NSL–KDD datasets. The pairs (norm,ϵ) define three different buffer configurations exploited in the evaluation of the learning algorithm.

	MNIST 0vs.1		MNIST 1vs.2		NSL–KDD		Buffer-Size
Norm	ϵ	N	ϵ	N	ϵ	N	
L_2	5.5	1712	6.3	1735	0.075	11195	Online setting 1 15% of the examples
L_1	55	1620	64	1699	0.12	11049	
$L_{0.5}$	7700	1679	9000	1775	0.35	11087	
L_2	6	1105	6.7	1133	0.15	8000	Online setting 2 10% of the examples
L_1	60	1165	70	1127	0.25	7723	
$L_{0.5}$	9000	1089	10500	1114	0.75	7987	
L_2	6.7	533	7.3	584	1	3850	Online setting 3 5% of the examples
L_1	70	569	80	559	1.25	3920	
$L_{0.5}$	11000	574	13000	529	5	3837	

Table 2 reports the classification performance of the OLapOCSVM algorithm on the MNIST and NSL–KDD datasets using the F1 score. The first group of results, referred to as *Batch setting*, corresponds to the (offline) batch training of the classifier on all the available training data. In the case of the NSL–KDD dataset it was not possible to train the classifier due to the large memory requirements needed to manage all the data. As a matter of fact, the proposed buffering strategy is not used and the training examples are not pruned. From a

Table 2. Accuracy of the OLapOCSVM algorithm on the MNIST and NSL–KDD datasets measured by the F1 score for each norm and ϵ setting as defined in Table 1. *Batch setting* corresponds to the (offline) batch training on the all the available data.

		MNIST		NSL–KDD		
	Norm	0vs.1	1vs.2	DoS	U2R	probing
Batch setting	L_2	0.998	0.950	-	-	-
	L_1	0.943	0.992	-	-	-
	$L_{0.5}$	0.648	0.990	-	-	-
Online setting 1	L_2	0.980	0.981	0.929	0.361	0.692
	L_1	0.954	0.987	0.923	0.353	0.716
	$L_{0.5}$	0.642	0.991	0.910	0.398	0.716
Online setting 2	L_2	0.973	0.980	0.929	0.361	0.687
	L_1	0.933	0.989	0.924	0.564	0.713
	$L_{0.5}$	0.895	0.984	0.913	0.391	0.720
Online setting 3	L_2	0.975	0.977	0.933	0.361	0.669
	L_1	0.965	0.974	0.930	0.413	0.712
	$L_{0.5}$	0.716	0.965	0.918	0.353	0.718

comparison of the results obtained in the three different on–line settings, it can be noticed that the F1 score decreases when using a larger tolerance in the data support approximation, but the performances degrade only slightly whereas the memory requirements are considerably reduced. In particular, there is not a significant performance difference with respect to the batch case, showing that the applied buffer policy is able to accurately approximate the original data distribution. Our results in the MNIST data are roughly comparable to the ones of [5], even if [5] exploits both positive and negative supervisions. We also tried to replicate the algorithm described in [5] using a OCSVM, but we were not able to obtain satisfactory performances, maybe due to difficulties in choosing an appropriate value for the learning rate used in the stochastic gradient descent.

The main advantage of the proposed approach is its capability to manage large datasets with reasonable time and memory requirements without a significant decrease of the performances. It does not require an accurate tuning of the learning rate [5], even if it may be tricky to optimize the tolerance parameter ϵ to have a good compromise between performance and resource requirements.

4 Conclusions

We presented On–line Laplacian One–Class SVM, a learning algorithm that updates a one–class classifier while new training data becomes available. In particular, the proposed buffer management policy allows the use of the algorithm both in on–line scenarios and in cases where a huge quantity of data is available. The results on the MNIST and NSL–KDD datasets show encouraging performances with a significant reduction of the storage requirements. Future work will consider an extension to other on–line semi–supervised learning algorithms, such as S3VM [7], SBRS [4], and the application of the buffering strategy to online

clustering by MEEs [10]. The proposed approach will be applied to computer vision tasks in a continuous learning framework.

Acknowledgements. We thank Marco Gori for the fruitful discussions in the development of the ideas presented in this paper. This research was partially supported by the research grant PRIN2009 Learning Techniques in Relational Domains and Their Ap- plications (2009LNP494) from the Italian MURST.

References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 420–434. Springer, Heidelberg (2000)
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. The Journal of Machine Learning Research 7, 2399–2434 (2006)
3. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Compstat. pp. 177–186 (2010)
4. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Bridging logic and kernel machines. Machine learning 86(1), 57–88 (2012)
5. Goldberg, A., Li, M., Zhu, X.: Online manifold regularization: A new learning setting and empirical study, pp. 393–407. Springer (2008)
6. Gretton, A., Desobry, F.: On-line one-class support vector machines. an application to signal segmentation. In: Proceedings of Acoustics, Speech, and Signal Processing, vol. 2, pp. II–709. IEEE (2003)
7. Joachims, T.: Transductive inference for text classification using support vector machines. In: Proceedings of ICML, pp. 200–209. Morgan Kaufmann (1999)
8. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. IEEE Transactions on Signal Processing 52(8), 2165–2176 (2004)
9. Melacci, S., Belkin, M.: Laplacian Support Vector Machines Trained in the Primal. Journal of Machine Learning Research 12, 1149–1184 (2011)
10. Melacci, S., Gori, M.: Unsupervised learning by minimal entropy encoding. IEEE Transactions on Neural Networks and Learning Systems 23(12), 1849–1861 (2012)
11. Muñoz-Marí, J., Bovolo, F., Gómez-Chova, L., Bruzzone, L., Camp-Valls, G.: Semisupervised one-class support vector machines for classification of remote sensing data. IEEE Trans. on Geoscience and Remote Sensing 48(8), 3188–3197 (2010)
12. Orabona, F., Castellini, C., Caputo, B., Jie, L., Sandini, G.: On-line independent support vector machines. Pattern Recognition 43(4), 1402–1412 (2010)
13. Radovanović, M., Nanopoulos, A., Ivanović, M.: Hubs in space: Popular nearest neighbors in high-dimensional data. The Journal of Machine Learning Research 9999, 2487–2531 (2010)
14. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. 13(7), 1443–1471 (2001)
15. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: Proceedings of the International Conference on Computational Intelligence for Security and Defense Applications, pp. 53–58 (2009)