On–line Laplacian One–Class Support Vector Machines

Nedo, Nodo, and Noccolo

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche
list e-mail here

To Remove: focus on online learning, why one-class (modular among classes, capable of managing class hierarchies as in KDDcup dataset), why laplacian (typically many unsupervised data in an online setting). Brief review of existing approaches (Goldberg).

Dataset description: MNIST and KDD. Experimental setup: different pruning strategies (L0.5, L1, L2, sliding window); best pruning strategy with laplacian; comparison with Goldberg. Advantages wrt Goldberg: no tuning of learning rate, policy for pruning with Euclidean metric.

Abstract. We propose an on–line manifold regularization algorithm that is able to work in scenario where the data arrives continuously over time and is not possible to store it before to learn the classifier. We present the On–line Laplacian One–Class SVM (OLapOCSVM) an algorithm that learns from positive labeled and unlabeled data in on–line fashion. The our approach is based on conjugate gradient descent on RKHS and therefore it receives the theoretical properties of standard convex programming technique. We present an efficient way to deal with the continuous incoming data by means of a buffer which management policy is based on the current estimate of the support of the input data distribution.
The experimental results show that OLapOCSVM reaches a level of performance comparable to batch algorithm LapOCSVM while it maintains the ability to work in on–line scenarios. The approach can be extended to the general semi–supervised learning setting and is an important step in direction of long learning real–world applications.

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

1 Introduction

Nowadays, in many applications, ranging from computer vision (a moving robot with a camera) to data mining (extract knowledge from a database), there is a great availability of data. In particular, many scenarios are typically on–line where the data arrives continuously during the time. The capabilities of the standard batch machine learning algorithms are limited by the available resources

both in term of time and memory. For this reasons, a great attention is focused on on–line machine learning algorithms that are able to work with never ending learning scenario [9]. The classifier has to be continuously improved and therefore the training process must be efficient as possible and to be able to adapt to change on the data distribution. Moreover, the real–world problems are naturally semi–supervised where limited quantity of labeled data and abundant quantity of unlabeled data are available [2]. In this direction has been proposed an online approach for manifold regularization applied to SVM [6]: the authors extend the idea of online SVM [9] to semi–supervised setting by exploiting the Laplacian of the data. They propose different strategies to take into account the time and memory requirements of the algorithm.
In many problems, in both supervised and semi–supervised setting, there is an asymmetry in the data distribution: that is an examples may belong to several classes and so can be labelled with different labels i.e. positive examples. At the same time, given a certain class, each examples for which the label for the class is not specified is considered as belonging to the rest of the classes (one or more classes) i.e. negative examples. This is a well–known problem for the standard SVM: one possible solution is the so called One–Class SVM (OCSVM) that is a density estimation algorithm sometimes known as novelty detection algorithm [14]. In this scenario, the algorithm tries to estimate the distribution from positive examples since that they are the unique available [11]. Moreover, the other appealing property of OCSVM is its capability to model class hierarchies. Two straightforward online extension of the density estimation algorithm are [9] and [7].
In this paper is proposed an approach called Online Laplacian One–Class SVM (OLapOCSVM) that exploits as baseline the Laplacian SVM (LapSVM) algorithm [10] and in particular its One–Class extension in on–line fashion: in detail, the continuous stream of data is faced by means a pruning strategy that decides if the new example has to be added to the buffer depending on the current estimate of the support of the input data distribution. The theoretical properties of convergence of the algorithm are guaranteed since that it exploits standard convex programming technique (conjugate gradient descent) [10] instead online convex programming technique (stochastic gradient descent) [4, 6].
Finally, the approach is not limited to the semi–supervised case analysed in this paper i.e. manifold regularization, but it is more general and can be employed with every semi–supervised learning algorithm.

2 Methods

The considered learning scenario consists in a stream of data for which supervisions are given only for some examples belonging to a class of interest. Hence the training algorithm goal is to develop the class model on–line while the data is made available, taking advantage of both supervised and unsupervised records. The first aspect to be taken into account is that only positive examples are provided in 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 minimizing the following functional with respect to the function $f : \mathcal{X} \to I\!R$ in a given RKHS $\mathcal{H}$ and the bias $b \in I\!R$,

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

where $\mathcal{L}^+$ is the set of supervised positive examples. 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.

In the on–line setting data is incrementally made available in time and the classifier should be able to provide its predictions at each time step. Hence, the learning algorithm should be designed to provide an optimal solution at each time step t, given the data seen up to that instant. However, since in general the horizon is infinite, it is unfeasible to collect all data up to t and to train the classifier on them. In fact, this approach would require an unbounded amount of memory and increasingly long training times for the classifier. Finally, retaining all the data history may degrade the performances in these cases in which there is a drift in the data distribution. Therefore, a classical approach to on–line learning is to exploit a finite size buffer that collects only part of the incoming data [9, 6]. This technique requires to define an appropriate policy to manage the buffer overflow. For 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 case of replacement, a simple criterion is to remove the oldest memorized example, thus implementing a sliding window of the input stream. A better solution is to consider also the significance of the examples for the classifier. For instance, a good strategy is to give priority to supervised examples with respect to unsupervised ones [6]. More advanced methods try to estimate the contribution of the new point or of the example to be removed to the accuracy of the estimated classifier [12]. In particular, 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 give 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 allow 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. 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 will consider two different configurations: a buffer with a fixed maximum size with a replacement policy in case of overflow based on the example age, with priority for supervised examples, referred to as *on–line buffer* **to remove**; a buffer whose insertions are managed by the pruning technique defined by the rule of eq. (2), referred to as *on–line pruning.*

Finally, also unsupervised examples available in the input stream can be exploited to provide additional information on the data distribution. By following the Laplacian SVM approach [2, 10] 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 $\mathcal{S} = \mathcal{L}^+ \cup \mathcal{U}$: the W is the adjacency matrix of the data graph, which entry in position i, j is w_{ij}, and D is the diagonal matrix with the degree of each node i.e. the element $d_{ii} = \sum_{j=1}^{n} w_{ij}$. Hence, the manifold regularization term is defined as

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

Note that, in general, several choice of $\|f\|_M^2$ are possible and that in the case of exponential weights for the adjacency matrix $E_{\mathrm{g}}[f] = f^T L f$ [2].
The learning objective is to minimize the following functional with respect to f and b:

$$E = E_{\mathrm{oc}}[f] + E_{\mathrm{g}}[f]. \tag{4}$$

If we consider the batch case, there are two fundamental differences between the Laplacian SVM approach and the proposed Laplacian OCSVM approach. The first is that the our approach make use only of positive examples and therefore the graph Laplacian will have a connections only among these examples and the unsupervised ones that are similar. A relevant quantity of unsupervised points that are similar to negative examples will have no connections on the Laplacian graph. The second is related to the computation of the adjacency matrix of the data graph W: the Laplacian SVM approach employs the k–Nearest Neighbour rule to evaluate the k examples $\mathbf{x}_j,\ j = 1, ..., k$ that are more similar to the given example $\mathbf{x}_i$ and then computes the corresponding edges in the graph as RBF kernel $w_{ij} = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_p}{2\sigma_L}\right)$ where $\|\mathbf{x}_i - \mathbf{x}_j\|_p$ is a distance between x_i and x_j and σ_L is an hyper parameter [2]. The proposed approach decides that an example $\mathbf{x}_j$ is similar to a given example $\mathbf{x}_i$ if their distance $\|\mathbf{x}_i - \mathbf{x}_j\|_p$ is under a predefined tolerance ϵ_L and computes the corresponding edges in the graph as RBF kernel $w_{ij} = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_p}{2\sigma_L}\right)$. In the batch case, this criteria is theoretically motivated by the fact that we want avoid to use the k–Nearest Neighbour rule as it is well–known that in high dimensional space suffers of several problems [3, 13]. If we consider the online case, the crucial issue is the efficiency of the computation of the Laplacian graph: the proposed criteria allows to compute incrementally the adjacency matrix W each time t that the buffer $\mathcal{B}_t$ is updated. The k–Nearest Neighbour does not permit the incremental computation of the adjacency matrix. Moreover, since that we use the same distance $\|\cdot\|_p$ for the policy of management of the buffer, the RBF kernel and for the Laplacian graph, the method remains efficient avoiding a lot of duplicated computations.

3 Experiments

We provide results on two data sets: MNIST[1] and NSL-KDD[2].

The MNIST is a very well-known digit classification dataset. Here we consider the same tasks as in [6], which employed two binary classification problems (digit 0 vs. 1 and 1 vs. 2): the key difference is that we only consider the positive class to train our one-class algorithm (that is, digit 0 for 0 vs. 1, and digit 1 for 1 vs. 2). For all our experiments we used the original training set and test set without any preprocessing except the normalization of the gray-level value of each pixel.

The NSL-KDD is an anomaly detection data set, built upon the original KDD Cup 1999 data set, but developed in order to solve several problems in the original data (e.g., see [15] for a detailed analysis). The NSL-KDD data set consists of examples describing network traffic in different timestamps, each one labelled with one of following classes: DoS, R2L, U2R, probing and normal (no attack). In our experiments we discarded the R2L class as it contains too few examples (0.07% on training set and 0.37% on the set). Each continuous attribute in the NSL-KDD data set was discretized into 10 bins, by selecting the quantization thresholds using a maximum entropy principle (each bin has to contain roughly the same number of examples).

For both data sets, the hyper parameters of the algorithms are selected by means of cross validation (1). The first 10% of the original training set has been split in two part: the first part, that consists of 66% of examples of which 10% are supervised, has been used for training while the second part, that consists of the remaining 34% of examples, is used for validation.

Dataset	Hyper parameter	Range	Meaning
	σ_{kernel}	$(3, ..., 21)$, in 3 steps	The width of the RBF kernel
	λ_r	$(10^{-1}, ..., 10^{-6})$, in 10^{-1} steps	The regularization parameter
MNIST	λ_m	$(0, 1, ..., 10^{-4})$, in 10^{-1} steps	The manifold regularization parameter
	$\sigma_{Laplacian}$	$(3, ..., 12)$ in 3 steps	The width of RBF kernel for edge weighting for adjacenc
	ϵ_L	$(3, ..., 12)$ in 3 steps	The tolerance parameter used for computing the adjacen
	σ_{kernel}	$(3, ..., 21)$, in 3 steps	The width of the RBF kernel
	λ_r	$(10^{-1}, ..., 10^{-6})$, in 10^{-1} steps	The regularization parameter
KDD	λ_m	$(0, 1, ..., 10^{-4})$, in 10^{-1} steps	The manifold regularization parameter
	$\sigma_{Laplacian}$	$(0.5, 1, 1.5, 2, 3)$	The width of RBF kernel for edge weighting for adjacenc
	ϵ_L	$(0.5, 1, 1.5, 2, 3)$	The tolerance parameter used for computing the adjacen

Table 1. The table reports the hyper parameters, their range and meaning for MNIST and KDD dataset.

[1] http://yann.lecun.com/exdb/mnist/

[2] http://iscx.ca/NSL-KDD/

Then, the algorithms has been trained on the second part of the original training set (90%) and evaluated on the original test set. In general, hyper parameters are hard to adjust when only positive labeled examples are available in the training set. In these situations is possible to compute only the true positive rate and therefore is not possible to compute the F1 score. In these cases different measure of performance evaluation have to be taken into account [11]. In our case this is not a problem since that the training set contains both positive and negative labeled samples: we use the one–vs–all approach for training the algorithm on both dataset.

Summarizing the three step procedure used is the following:

- Hyper parameter selection. In this step, the hyper parameters of the classifier are tuned on the validation dataset as describe before.
- Classifier training. Using the optimal set of hyper parameters found in the previous step, the final classifier is trained using all the available labeled and unlabeled examples in the training set.
- Classifier testing. The trained classifier is used to predict the test set, the confusion matrix is computed and the performance are measured by F1 score.

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

Table 2. The table reports the tolerance parameter ϵ and the corresponding number of examples N for each norm ($L2$,$L1$ and $L0.5$) on MNIST and KDD datasets.

Three online setting are determined by varying the tolerance ϵ in such a way different number of examples N of the original training set is maintained (table 2). In particular, the table shows that L_1 and $L_{0.5}$ norms need of a high values of the tolerance parameter to produce the same number of examples of the L_2 norm. This is motivated by the fact that the data results more sparse as long as the p of L_p decreases [1].
On MNIST, the performance of the proposed approach OLapOCSVM are good in all three online setting (table 3): in particular, there is not great difference w.r.t. the batch LapOCSVM that uses the entire dataset. Similar results are achieved on KDD dataset (table 4): note that in this case is not possible to train the batch LapOCSVM a due to the dimension of the entire dataset. The results

do not show a norm that is better to the others: although the L_2 norm seems to perform better on the MNIST dataset while the L_1 norm seems to perform better on the KDD dataset. These results are in contrast with [1] since that the **motivate it**.
We have tried to replicate the approach [6] since that no code is available and we get not encouraging results: in detail, we were not able to reproduce the performance reported on the paper on MNIST dataset. The main problem is related on adjusting the value of the learning rate: their approach is based on stochastic gradient descent which performance are highly dependent on this parameter [4]. The main advantage of the our approach is its capability to treat big dataset, as KDD, with reasonable time and memory requirements and with good level of performance. Moreover, it does not require the tuning of the learning rate parameter [10] as the approach of [6]. One drawback of the our approach is on determining the tolerance parameter ϵ in such a way it gives good compromise between performance and resource requirements.

	Norm	0vs.1	1vs.2
Original dataset	L_2	0.998	0.950
	L_1	0.943	0.992
	$L_{0.5}$	0.648	0.990
Online setting 1 (15% examples)	L_2	0.980	0.981
	L_1	0.954	0.987
	$L_{0.5}$	0.642	0.991
Online setting 2 (10% examples)	L_2	0.973	0.980
	L_1	0.933	0.989
	$L_{0.5}$	0.895	0.984
Online setting 3 (5% examples)	L_2	0.975	0.977
	L_1	0.965	0.974
	$L_{0.5}$	0.716	0.965

Table 3. The table reports the performance, for each norm (L_2,L_1 and $L_{0.5}$), in term of F1 score for the OLapOCSVM algorithm on MNIST dataset.

4 Conclusions

We present the Online Laplacian One–Class SVM that is a manifold learning algorithm based on convex optimization on RKHS. The presented policy of management of the buffer allows to make practical the usage of the algorithm in on–line scenarios where the data flow is continuous and in batch scenarios where huge quantity of data is available. The results show promising performance on two popular dataset MNIST and KDD.
Future work will be carried on two directions. On theoretical side, we will study how to apply the key properties of the OLOCSVM to the general online semi–supervised learning algorithms like standard S3VM [8] and SBRS [5]. On empir-

	Norm	DOS	U2R	probing	Attack
Online setting 1 (15% examples)	L_2	0.929	0.361	0.692	0.883
	L_1	0.923	0.353	0.716	0.883
	$L_{0.5}$	0.910	0.398	0.716	0.883
Online setting 2 (10% examples)	L_2	0.929	0.361	0.687	0.883
	L_1	0.924	0.564	0.713	0.883
	$L_{0.5}$	0.913	0.391	0.720	0.883
Online setting 3 (5% examples)	L_2	0.933	0.361	0.669	0.883
	L_1	0.930	0.413	0.712	0.883
	$L_{0.5}$	0.918	0.353	0.718	0.883

Table 4. The table reports the performance, for each norm (L_2,L_1 and $L_{0.5}$), in term of F1 score for the OLapOCSVM algorithm on KDD dataset.

ical side, we will perform a deep analyses on the ability of the algorithm to adapt to real–world application: in particular, we are interested on investigating the performance on computer vision application as a moving robot with a camera.

5 Acknowledgements

References

1. Aggarwal, C., Hinneburg, A., Keim, D.: On the surprising behavior of distance metrics in high dimensional space. Database TheoryICDT 2001 pp. 420–434 (2001)
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. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful? Database TheoryICDT99 pp. 217–235 (1999)
4. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Compstat. vol. 2010, pp. 177–186 (2010)
5. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Bridging logic and kernel machines. Machine learning 86(1), 57–88 (2012)
6. Goldberg, A., Li, M., Zhu, X.: Online manifold regularization: A new learning setting and empirical study. Machine Learning and Knowledge Discovery in Databases pp. 393–407 (2008)
7. Gretton, A., Desobry, F.: On-line one-class support vector machines. an application to signal segmentation. In: Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on. vol. 2, pp. II–709. IEEE (2003)
8. Joachims, T.: Transductive inference for text classification using support vector machines. In: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-. pp. 200–209. MORGAN KAUFMANN PUBLISHERS, INC. (1999)
9. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. Signal Processing, IEEE Transactions on 52(8), 2165–2176 (2004)
10. Melacci, S., Belkin, M.: Laplacian Support Vector Machines Trained in the Primal. Journal of Machine Learning Research 12, 1149–1184 (March 2011)

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. Geoscience and Remote Sensing, IEEE Transactions on 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 (Jul 2001), `http://dx.doi.org/10.1162/089976601750264965`
15. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications. pp. 53–58. CISDA'09, IEEE Press, Piscataway, NJ, USA (2009), `http://dl.acm.org/citation.cfm?id=1736481.1736489`