Choosing the Best Algorithm for an Incremental On-line Learning Task

Viktor Losing[1][2] Barbara Hammer[2] and Heiko Wersing[1]

1- HONDA Research Institute Europe GmbH,
Carl-Legien-Str. 30, 63065 Offenbach - Germany

2- Bielefeld University, Universitaetsstr. 25, 33615 Bielefeld - Germany

Abstract. Recently, incremental and on-line learning gained more attention especially in the context of big data and learning from data streams, conflicting with the traditional assumption of complete data availability. Even though a variety of different methods are available, it often remains unclear which of them is suitable for a specific task and how they perform in comparison to each other. We analyze the key properties of seven incremental methods representing different algorithm classes. Our extensive evaluation on data sets with different characteristics gives an overview of the performance with respect to accuracy as well as model complexity, facilitating the choice of the best method for a given application.

1 Introduction

Attention in the field of incremental learning has grown drastically in recent years which is caused by two major developments. The main reason is the rapidly increasing amount of data accumulation in fields such as health monitoring, social networks, financial transactions or personalized web services [1], [2], demanding for continuous large-scale, real-time processing. Another trend is to incorporate individual adaptation to customer habits and environments within smart home products [3], [4] requiring skills such as efficient learning from few data.
A lot of ambiguity is involved regarding the definition of incremental and on-line learning in the literature. Some authors use them interchangeably, while others distinguish them in different ways. Additional terms such as lifelong- or evolutionary learning are also used synonymously. We define an incremental learning algorithm as one that generates on a given stream of training data $x_1, x_2, ..., x_t$ a sequence of models $M_1, M_2, ..., M_t$ and thereby meets the following criteria:

- M_{t+1} adapts gradually based on M_t without a complete retraining.
- It preserves previously acquired knowledge and does not suffer from the effect of catastrophic forgetting.
- It has a limited stream memory.

We specify on-line learning algorithms as incremental learning algorithms which are additionally bounded in model complexity and run-time, capable of endless/lifelong learning on a device with restricted resources.
A variety of interesting incremental learning algorithms have been published but there is a lack of in-depth comparative studies. In this paper we contribute to fill this gap by analyzing the core attributes of seven popular methods. Experiments on diverse datasets assess their performance and provide guidance on their

applicability for specific tasks. We focus solely on the classification of stationary datasets (i.e. we assume the stream $x_1, x_2, ...$ is i.i.d.) and not yet analyze the methods in context of concept drift. A recent overview of methods especially designed to deal with non-stationary environments is given in [5].

2 Algorithms

In the following we briefly describe all tested methods including Bayesian -, linear -, instance based models as well as ensembles and neural networks.
Incremental Support Vector Machine (ISVM), introduced in [6], is the only exact incremental version of the SVM. A limited number of examples, so called "candidate vectors", which could be promoted to support vectors in the future, is additionally maintained.
On-line Random Forest (ORF)[7] is an incremental version of the Extreme Random Forest. A predefined number of trees grow continuously by adding splits whenever enough samples are gathered within one leaf. Tree ensembles are very popular, due to their high accuracy, simplicity and parallelization capability.
Incremental Learning Vector Quantization (ILVQ) extends the Generalized Learning Vector Quantization (GLVQ) to a dynamically growing model by continuous insertion of new prototypes. We introduced a superior prototype placement strategy in [8] minimizing the loss on a window of recent samples. Metric learning, as described in [9], can also be applied.
Learn++ (LPP)[10] processes incoming samples in chunks with a predefined size. For each chunk an ensemble of base classifiers is trained and combined through weighted majority voting to an "ensemble of ensembles". Chunk-wise trained models have by design an adaption delay depending on the chunk size.
Incremental Extreme Learning Machine (IELM) reformulates the batch ELM least-squares solution into a sequential scheme [11]. As the batch version it drastically reduces the training complexity by randomizing the input weights. The network is static and the number of hidden neurons has to be predefined.
Gaussian Naive Bayes (GNB) fits one axis-parallel Gaussian distribution per class and uses them as likelihood estimation in the Naive Bayes algorithm [12]. The algorithm is lossless, i.e. it generates the same model as the corresponding batch algorithm, and independent from the training order.
Stochastic Gradient Descent (SGD) is an efficient method for learning a linear, discriminative model by minimizing a convex loss function. Revived recently in the context of large-scale learning [13] it performs especially well for sparse, high-dimensional data as often encountered in the domain of text classification.

3 Experiments

We used the implementations of the Scikit-learn package [14] for SGD and GNB. All the others are derived from the code of the respective authors. Only publicly available datasets (see [15], [16]), predefining a fixed train-test-split, were used to enable reproducibility and comparability of our results. Table 1 gives the main attributes of the selected datasets. Artificial and real world problems are included, differing widely in the number of classes, instances and dimensions. Links to all implementations and datasets are available at `https://github.com/vlosing/Online-learning`.

3.1 Hyperparameter setting

The model selection is varying in complexity depending on the parameter amount and type. Tree based models usually perform well out of the box, whereas the ISVM or ILVQ require an accurate, dataset dependent configuration of multiple parameters to deliver good results. The ISVM was solely paired with the RBF kernel. We used the metric learning of ILVQ only for datasets with up to 250 dimensions. The GNB algorithm is parameterless, hence no tuning is required at all. We minimize the hinge loss function with SGD using the default setting of Scikit-learn and adjust only the learning rate. LPP requires the number of base classfier per chunk as well as the parameters of the base classifier itself (non-parametric Classification and Regression Trees in our case).

Dataset	#Train	#Test	#Feat.	#Class
Border	4000	1000	2	3
Overlap	3960	990	2	4
Letter	16000	4000	16	26
Outdoor	2600	1400	21	40
COIL	1800	5400	21	100
DNA	1400	1186	180	3
USPS	7291	2007	256	10
Isolet	6238	1559	617	26
MNist	60000	10000	784	10
Gisette	6000	1000	5000	2

Table 1: The evaluated datasets.

3.2 Measure of model complexity

The algorithm implementations vary in the written programming languages as well as their efficiency. Therefore, we do not compare training- and run-time but instead we measure the model complexity by counting the number of parameter required for the representation. This enables a comparison with respect to memory consumption. However, the models are fundamentally different so that this measure, even though there is some correlation, should not generally be equated with training- or run-time. For instance, the $\mathcal{O}(\log l)$ run-time of tree based models, l being the number of leaves, make them incredible fast in comparison to instance based ones ($\mathcal{O}(d * i)$, for i instances (e.g. prototypes, support vectors) with d dimensions). Another example are the Extreme Random Trees deployed by the ORF algorithm. By introducing random splits they drastically reduce the training-time, but also require larger trees[17]. We rather use this measure as an indicator to decide whether an algorithm struggles (unreasonable high amount of parameters) or is especially suited (sparse representation paired with high accuracy) for a given task.

3.3 Results

The evaluation of GNB, ORF and SGD is straightforward since these access consecutively only the current training example. But methods such as ISVM

	ISVM	ORF	ILVQ	LPP	IELM	SGD	GNB	ISVM	ORF	ILVQ	LPP	IELM	SGD	GNB
Border	**99.4**	97.6	96.8	96.5	96.0	35.5	96.4	797	3.7k	193	1.9k	750	**9**	12
Overlap	82.2	**83.7**	83.0	81.7	83.4	67.6	66.4	10k	2.3k	235	1.9k	900	**12**	16
Letter	**97.0**	93.2	93.9	87.0	70.0	56.4	63.4	131k	168k	16k	51k	8.4k	**442**	832
Outdoor	70.9	**71.0**	66.9	68.5	70.9	23.2	62.2	40k	8.8k	11k	6.2k	12k	**880**	1.7k
COIL	**96.5**	92.9	94.3	89.2	91.5	12.4	92.4	58k	61k	18k	9.2k	36k	**2.2k**	4.2k
DNA	**94.9**	89.6	92.1	90.5	88.8	93.0	89.1	237k	5.0k	33k	1.6k	55k	**543**	1.0k
USPS	**95.4**	92.5	91.4	90.3	92.1	89.0	75.8	710k	33k	15k	9.6k	106k	**2.6k**	5.1k
Isolet	**96.2**	92.5	92.0	90.0	91.9	91.5	80.1	2.8M	31k	21k	**12.0k**	322k	16k	32k
MNist	-	94.3	**94.8**	92.4	89.1	86.0	51.2	-	111k	315k	73k	397k	**7.9k**	16k
Gisette	**98.0**	94.6	93.0	94.2	91.4	93.1	71.7	7.0M	4.7k	263k	**2.5k**	2.5M	5.0k	20k
∅	**92.3**	90.2	89.8	88.0	86.5	64.7	74.9	1.2M	43k	69k	17k	344k	**3.5k**	8.0k
∅ Rank	**1.4**	2.3	3.1	4.4	4.5	5.6	6.1	6.6	5.6	4.1	3.3	5.2	**1.3**	2.5

Table 2: Test accuracy (left) and model complexity (right) after training, measured by the number of parameters and averaged over 10 repetitions.

and ILVQ store additionally a window of recent samples or require chunk-wise training, as LPP and IELM[1] do. In both cases, results depend on the window-/chunk size. Therefore, we tested several sizes of up to 1000 samples and chose the one giving the highest accuracy for the respective algorithm. All methods were trained single-pass, in the same order after initial shuffling.
Table 2 shows the accuracies and corresponding model complexities at the end of training. The ISVM achieves in average the highest accuracies, often with a large margin, but at the expense of having by far the most complex model. The large amount of parameters is partly due to the fact that the model is designed to discriminate 2 classes, resulting in multiple SVMs to perform schemes such as one vs. all in case of more classes. Another reason is the linear growth of support vectors with the amount of samples. The model gets exceedingly complex for noisy or overlapping datasets such as *Isolet* or *Overlap*. Its high training-complexity, resulting from the computation and incremental update of the inverse kernel matrix, prevents an application for datasets consisting of substantially more than 10000 samples such as *MNist*[2]. The instance based ILVQ constructs a far sparser model and achieves high accuracies throughout all datasets. As expected, tree based models require a comparably large amount of parameter for low dimensional data but are eminently efficient in high dimensional spaces, due to their compressing representation. The ORF has the second highest accuracies and constantly beats LPP. One explanation, already noticed in [18], is that LPP trains each base classifier with samples of only one chunk. Therefore, the knowledge integration across chunks is limited since it is exclusively established by the weighting process. Furthermore, the ORF benefits more from the sub-linear tree complexity because it generates a few deep trees instead of the numerous, shallow ones by LPP. The SGD model uses the fewest parameters and performs especially well for high dimensional data. However, it struggles by design with

[1]IELM requires for the initialization at least as many samples as it has hidden neurons but afterwards it can be updated after each sample.

[2]We canceled the training after one day.

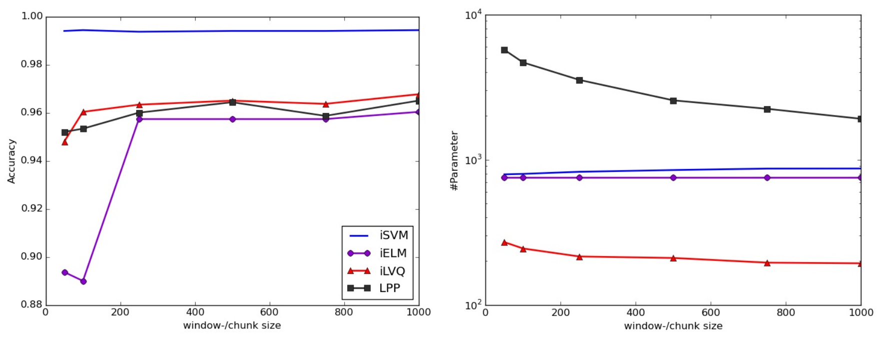

Fig. 1: Influence of the window-/chunk size on the accuracy (left) and model complexity (right) for dataset *Border*.

.

non-linear separability as it is the case for the *Border* dataset, or whenever a small amount of examples is available per class (*COIL*). The last rank of GNB obscures the fact that it performs reasonably well without severe hiccups, incorporating a simple and sparse model. Nonetheless, major drawbacks are its restriction to unimodal distributions as well as its independence assumption.
The typical effects of different window-/chunk sizes are shown in fig. 1 exemplary for the *Border* dataset. Usually the algorithms do benefit from an increased window-/chunk size. For instance, a larger window enables the ILVQ to find better positions for new prototypes and the ISVM to miss less support vectors. Simultaneously, the model complexity of ILVQ is reduced since the insertion rate is coupled with the training-error. In case of LPP, however, larger chunks reduce the number of base classifiers but at the same time each of them is trained on more training data, requiring a balancing of these criteria.

3.4 Restriction of the overall classifier complexity

Methods as SGD, NB and IELM are on-line algorithms since they are constant in their complexity. ILVQ and LPP can be easily restricted to a certain limit by strategies such as removing the "worst“ prototype/classifier [19], [20]. In case of the ISVM, however, it is less trivial. Even though methods as [21] do reduce the number of support vectors, there is to the best of our knowledge no practical method to bound them strictly. This applies to a lesser degree also for the ORF. It learns by growing its trees continuously. Therefore, a depth reduction or pruning mechanism would be necessarily at some point.

4 Conclusion

We analyzed the most common algorithms of incremental learning on diverse, stationary datasets. Regarding the results, the ISVM delivers usually the highest accuracy at the expense of the most complex model. Its training time is tolerable for tasks consisting of up to 10000 samples. The ORF performs slightly worse but has a very fast training- and run-time. However, its model as well as those of the ISVM grows linearly with the number of samples and cannot be limited in a straightforward way. Therefore, both algorithms are not suited for learning in endless streams in contrast to all remaining methods, having either a constant or easily boundable complexity. The ILVQ offers an accurate and sparse alternative to the ISVM. LPP is quite flexible since the base classifier can be arbitrary selected, however, it may struggle by its limited knowledge integration across chunks. Tree based models are especially suitable for high dimensional data because of their compressed representation as well as their sub-linear run-time, which does not depend on the number of dimensions. The linear model of SGD is also a good choice for large-scale learning in high dimensional spaces.

References

[1] Lohr S. The age of big data. *New York Times*, 2012.

[2] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.

[3] Rayoung Yang and Mark W. Newman. Learning from a learning thermostat: Lessons for intelligent systems for the home. UbiComp '13, pages 93–102. ACM, 2013.

[4] Berardina De Carolis, Stefano Ferilli, and Domenico Redavid. Incremental learning of daily routines as workflows in a smart home environment. *ACM*, 4(4):20:1–20:23, 2015.

[5] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *Computational Intelligence Magazine*, 10(4):12–25, 2015.

[6] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Proc. NIPS*, 2001.

[7] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *ICCV Workshops 2009 IEEE 12th International Conference on*, 2009.

[8] V. Losing, B. Hammer, and H. Wersing. Interactive online learning for obstacle classification on a mobile robot. In *IJCNN 2015*, pages 1–8, July 2015.

[9] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Comput.*, 21, December 2009.

[10] R. Polikar, L. Upda, S.S. Upda, and V. Honavar. Learn++: an incremental learning algorithm for supervised neural networks. *SMC*, 31(4):497–508, Nov 2001.

[11] N-Y. Liang, G-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *NN*, 17(6):1411–1423, 2006.

[12] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. Technical report, Stanford, CA, USA, 1979.

[13] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] M. Lichman. UCI machine learning repository, 2013.

[16] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[17] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[18] Haibo He, Sheng Chen, Kang Li, and Xin Xu. Incremental learning from stream data. *Neural Networks, IEEE Transactions on*, 22(12):1901–1914, 2011.

[19] Mihajlo Grbovic and Slobodan Vucetic. Learning vector quantization with adaptive prototype addition and removal. In *IJCNN 2009*, pages 994–1001. IEEE, 2009.

[20] Ryan Elwell and Robi Polikar. Incremental learning in nonstationary environments with controlled forgetting. In *IJCNN 2009*, pages 771–778. IEEE, 2009.

[21] Tom Downs, Kevin E. Gates, and Annette Masters. Exact simplification of support vector solutions. *J. Mach. Learn. Res.*, 2:293–297, March 2002.