ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

Certainty-based Prototype Insertion/Deletion for Classification with Metric Adaptation

Lydia Fischer[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, Universitätsstr. 25, 33615 Bielefeld - Germany

Abstract. We propose an extension of prototype-based classification models to automatically adjust model complexity, thus offering a powerful technique for online, incremental learning tasks. The incremental technique is based on the notion of the certainty of an observed classification. Unlike previous work, we can incorporate matrix learning into the framework by relying on the cost function of generalised learning vector quantisation (GLVQ) for prototype insertion, deletion, as well as training. In several benchmarks, we demonstrate that the proposed method provides comparable results to offline counterparts and an incremental support vector machine, while enabling a better control of the required memory.

1 Motivation

Machine learning methods such as learning vector quantisation (LVQ) or support vector machines (SVM) provide state of the art classification schemes for automated data analysis [1, 2, 3]. The techniques are mostly used in offline scenarios where a suitable model complexity can be adjusted based on cross-validation. This procedure becomes prohibitive for big or streaming data sets and life-long learning, where data cannot be inspected at once, and an appropriate model complexity cannot be determined prior to training. In this setting, online, incremental, or streaming algorithms are of interest, which are capable of adapting its model complexity while training based on the observed data [4].

For SVM, incremental variants have been proposed which inspect only one data point at a time, but require extensive storage space due to a growing number of support vectors[1] [5, 6, 7]. For LVQ, a few online heuristics have been proposed: some only incorporate new training data [8], while others adjust the number of prototypes either by error-based insertion only [9, 10, 11], or dynamic prototype deletion and insertion [12, 13]. Unsupervised counterparts are for example the growing neural gas [14] and extensions thereof. These techniques, however, are based on heuristic grounds rather than a cost function. Further, no technique dynamically adjusts the model complexity and the underlying metric.

In this article we introduce an incremental LVQ method which inserts and deletes prototypes based on a certainty measure strongly related to the GLVQ costs [15, 16] and an analysis of its interaction with metric adaptation. This method allows a fast adaptation to new training data by prototype insertion,

*BH gratefully acknowledges funding by the CITEC center of excellence. LF acknowledges funding by the CoR-Lab Research Institute for Cognition and Robotics and gratefully acknowledges the financial support from Honda Research Institute Europe.

[1]We will use the online SVM code at: `http://www.isn.ucsd.edu/svm/incremental/`

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

respecting noise or overlaps by prototype deletion. The method reaches results comparable to offline variants or the incremental SVM [5], using less memory.

2 Learning Vector Quantization

Assume υ training data points $\mathbf{x}_i$ with class label y_i such that $(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, \ldots, C\}$. A LVQ classifier consists of a set of prototypes $W = \{\mathbf{w}_j \in \mathbb{R}^n\}_{j=1}^k$ equipped with class labels $c(\mathbf{w}_j) \in \{1, \ldots, C\}$. A given point $\mathbf{x}_i$ is classified according to the label of the closest prototype, the best matching unit (BMU), as measured in the squared Euclidean distance $\|\mathbf{x} - \mathbf{w}\|^2$ or variants.

Given training data, generalised LVQ (GLVQ) [17] optimises the location of prototypes by means of a stochastic gradient descent on the cost function

$$E = \sum_{i=1}^{\upsilon} \Phi((d^+(\mathbf{x}_i) - d^-(\mathbf{x}_i))/(d^+(\mathbf{x}_i) + d^-(\mathbf{x}_i))) \tag{1}$$

where Φ is a monotonic increasing function, e. g. the logistic function. $d^{\pm}$ is the distance of a data point $\mathbf{x}_i$ to the closest prototype $\mathbf{w}^{\pm}$ of the correct/incorrect class. A generalisation of the GLVQ towards a general quadratic form $(\mathbf{x} - \mathbf{w}_j)^T \Lambda (\mathbf{x} - \mathbf{w}_j)$ with positive semi-definite matrix Λ has been proposed under the acronym GMLVQ [1]. A local version thereof is the local GMLVQ (LGMLVQ) [1] where every prototype $\mathbf{w}_j$ has its own local metric $d_i(\mathbf{x}, \mathbf{w}_j) = (\mathbf{x} - \mathbf{w}_j)^T \Lambda_j (\mathbf{x} - \mathbf{w}_j)$. This cost function strongly correlates to the classification error since a data point is classified correctly iff the nominator of the cost function is below zero.

3 Incremental online LVQ

We introduce an incremental online learning for LVQ (ioLVQ) which can be used in combination with any of GLVQ, GMLVQ, or LGMLVQ. Starting with an empty set of prototypes, new prototypes are inserted or deleted on demand. Existing ones are adapted according to the standard GLVQ scheme, possibly including matrix learning. Hence the number of prototypes varies automatically to mirror the complexity of the observed training data. Note that the term

$$\mathrm{RelSim}(\mathbf{x}) = (d^-(\mathbf{x}) - d^+(\mathbf{x}))/(d^-(\mathbf{x}) + d^+(\mathbf{x})) \tag{2}$$

relates to the summands of the GLVQ costs and can be interpreted as a certainty of the classification: It provides values $\mathrm{RelSim}(\mathbf{x}) \in (-1, 1)$, where values near 0 indicate high uncertainty, high values near 1 indicate a high certainty, and values below 0 indicate a wrong classification since $d^+ > d^-$. It has been investigated recently that this value can serve as an efficient approximation of a confidence for classification with rejection [16]. In the following, this measure serves as signal which changes of prototypes likely decrease the GLVQ costs. There are three mechanisms to self-adjust the model complexity based on the observed data:

New classes insertion strategy taken from [13]: Each training point $(\mathbf{x}, y)$ with $y \neq c(\mathbf{w}_j), \forall\, j$, i. e. a new class, is directly used as new prototype with label y reducing the costs for class y.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

Prototype insertion: Wrong classifications increase the costs (1), hence lowering their number decrease the costs, provided the remainder is not greatly affected. Based on an idea from [9, 11], wrongly classified training data are stored in a set S with maximum storage capacity $g_{\max}$. Once $|S| = g_{\max}$, for each class p for which errors are stored (i. e. $\exists\, i : p = y_i \wedge \mathbf{x}_i \in S$) a prototype with label p is introduced. We determine its position to minimise costs, i. e. the point $(\mathbf{x}_i, p)$ in the set with lowest $\mathrm{RelSim}(\mathbf{x}_i)$ is chosen as prototype position. Note that this also corresponds to a high degree of uncertainty, that means a potentially useful placement. After this insertion, the set S is cleared, i. e. $S := \emptyset$.

Prototype deletion: Unbounded prototype insertion can generate prototypes with noisy Voronoi cells, hence they harm more than they use. We remove prototypes based on their contribution to the costs (1), mimicking the idea underlying [12] that counts correct vs. wrong classifications as score. Here, every prototype $\mathbf{w}$ is accompanied by a parameter $\eta(\mathbf{w})$, initialised with zero, that sums up the certainty of the classifications of points in its Voronoi cell. Hence a presentation of a data point $(\mathbf{x}, y)$ with BMU $\mathbf{w}_l$ leads to the update:

$$\eta(\mathbf{w}_l) := \eta(\mathbf{w}_l) + \mathrm{RelSim}(\mathbf{x}) \ . \tag{3}$$

The value $\eta(\mathbf{w}_I)$ is negative if and only if, on average, the prototype accounts for more wrongly classified points than correct ones, weighted by their certainty. In periods of r_{num} seen training data points, prototypes $\mathbf{w}$ with $\eta(\mathbf{w}) < 0$ are removed since their deletion directly improves the GLVQ costs.

4 Experiments

Unless stated otherwise, we consider randomly ordered data presented in a single pass, like in a streaming setting, using constant learning rates. Evaluation is based on ten repeats of a 10-fold cross validation. Considered data sets are:

- Image: The image segmentation data set contains 2310 data points with 19 real-valued image descriptors. The data represents small patches from outdoor images with 7 different, balanced classes like grass, cement, etc. [18].
- Coil: The Columbia Object Image Database Library contains gray scaled images of 20 objects [19]. Rotating each object in 5° steps, provides 72 images per object. The data set has 1440 vectors with 16384 dimensions that are reduced with PCA [20] to 30.
- Outdoor[2]: The outdoor obstacle data set contains 4000 data points that contain the 21 values of a 6 bin rg chromaticity diagram. The data points belong to 40 objects such as dog, red ball, book, etc. lying in grass.

Incremental parameters: We first analyse the effect of the parameters $g_{\max}$ and r_{num} of ioGLVQ without metric learning (table 1, first half) since these parameters have a crucial influence of how the model deals with the stability/plasticity dilemma like the parameters of an unsupervised counterpart which is discussed

[2]Thanks to Viktor Losing for providing this real world data set.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Data	Err	$\|W\|$	Err	$\|W\|$	Err	$\|W\|$	Err	$\|W\|$
$g_{\max}$	20		50		80		100	
Image	**13.3**	**103.1**	20.1	62.6	23.6	48.2	25.9	42.4
Coil	**4.9**	**73.0**	**8.7**	**58.6**	**10.8**	**50.7**	**13.8**	**43.7**
Outdoor	**11.8**	**376.1**	**13.1**	**332.4**	**14.1**	**294.8**	**15.6**	**268.6**
r_{num}	60		150		250		400	
Image	**28.6**	**22.6**	**22.8**	**37.3**	**20.4**	**42.8**	**18.4**	**51.9**
Coil	**16.8**	**38.4**	**12.2**	**48.5**	10.9	52.1	**9.0**	**57.3**
Outdoor	**30.9**	**66.7**	**26.7**	**97.2**	**23.1**	**136.9**	**20.3**	**155.9**
global	a) random		b) unit matrix		c) mixture		d) trained matrix	
Image	16.3	21.9	17.2	21.6	16.4	22.2	13.0	17.9
Coil	11.6	30.4	11.5	30.6	11.1	29.6	5.5	23.1
Outdoor	18.6	179.5	16.2	184.1	19.2	180.6	13.7	194.7
local	a) random		b) unit matrix		c) mixture		e) next proto	
Image	17.5	16.6	17.0	16.5	17.6	16.9	16.7	15.8
Coil	3.2	23.9	3.2	23.5	3.7	24.1	3.0	24.3
Outdoor	19.9	186.2	17.9	191.3	20.1	187.0	19.9	185.8

Table 1: We report the average test errors Err and the average prototype number $|W|$. First part: Effect of the parameter $g_{\max}$ without removing and of the parameter r_{num} with a fixed $g_{\max} = 20$ (ioGLVQ). The marked results lie on the Pareto-front. Second part: Effect of the initialisation of the global/local metric.

in [21]. The first block shows results varying $g_{\max}$ only without prototype deletion ($r_{\text{num}} = \infty$). The parameter $g_{\max}$ controls the speed of prototype insertion, with small values accounting for a fast reacting but possibly noise-fragile system. A prototype removal strategy makes the system more robust with respect to noise.

The second block of table 1 shows the effect of r_{num} for a fixed $g_{\max}$. Changing r_{num} to high values increases the number of prototypes. Too small values of r_{num} prohibit a sufficient adaptation of prototype positions, and prototypes are merely replaced. The parameters $g_{\max}$ and r_{num} control the trade off between the error rate and the number of prototypes. In the first half of table 1, Pareto-optimal solutions with respect to these two criteria are marked in bold. Based on these findings, in the following, we choose $g_{\max} \approx 40$ and $r_{\text{num}} \approx 0.1 \cdot$ data set size.

Compatibility with metric learning: We analyse the effect of metric learning for ioLVQ using several initialisation methods for the metric: a) random values in (-1,1), b) the unit matrix, c) a unit diagonal and random elements otherwise d) initialisation with pre-trained global matrix obtained from a previous model, e) initialisation with local matrix of the next prototype from the same class. The second part of table 1 shows the results. Clearly, differences are only minor. We will use initialisations c) respectively e) as robust ones in the following.

Comparative Evaluation: We show three experimental settings: 1) a single streaming pass through the data; 2) multiple streaming passes (Image: 70, Coil: 120, Outdoor: 150; the number of passes is adjusted according to the convergence speed of batch LVQ); these results are compared with an incremental SVM [5]; 3) batch versions of the LVQ approaches, where the method is initialised with one prototype per class (Image, Coil) and four prototypes per class (larger values hardly influence the classification accuracy), this is compared with a batch SVM

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Data	Err	$\|W\|$	Err	$\|W\|$	Err	$\|W\|$	Err	$\|SV\|$
1)	GLVQ		GMLVQ		LGMLVQ			
Image	20.4	42.8	**16.3**	**21.9**	**15.9**	**23.5**		
Coil	9.0	57.3	11.1	29.6	**3.0**	**24.3**		
Outdoor	**20.3**	**155.9**	**16.2**	**184.1**	17.9	191.3		
2)	GLVQ		GMLVQ		LGMLVQ		incremental SVM	
Image	16.6	70.1	4.8	58.3	**3.4**	**55.0**	4.2	611.4
Coil[3]	3.0	83.6	0.7	67.1	**0.0**	**36.1**	0.7	1546.3
Outdoor	**16.6**	**241.9**	**14.8**	**255.8**	16.7	245.2	16.5	446.3
3)	GLVQ		GMLVQ		LGMLVQ		batch SVM	
Image	20.9	7	9.9	7	*5.2*	7	2.8	265.8
Coil	9.9	20	3.8	20	*0.8*	20	0.0	773.8
Outdoor	17.7	160	*12.6*	160	18.5	160	4.7	1537.4

Table 2: Comparison of the online and batch version of LVQ, batch SVM [22] and incremental SVM [5]. For SVM, multiple classes are addressed by one vs. rest encoding. We display the average test error Err and the average number of prototypes $|W|$, support vectors $|SV|$. For settings 1) and 2) Pareto optima are set boldface. In setting 3) the best error rate (omitting batch SVM) is set italic.

[22]. In 1) and 2) learning rates are constant while in 3) the learning rates are annealed (table 2). The learning rates (prototypes, metric) used in ioLVQ are roughly one magnitude smaller than in the batch version.

Learning a metric improves the performance and lowers the demand of prototypes, except for the outdoor data set. The reached performances are better than the performances of the batch counterparts for LVQ (except outdoor: GMLVQ) but the ioLVQ models use more prototypes than the batch models, in the worst case (image: GLVQ) ten times more. In general, this higher number of prototypes is acceptable since the number is still mostly in the same order of magnitude. The results of ioGMLVQ are comparable to the results of an incremental SVM, while ioLGMLVQ is even better for (Image, Coil).

5 Conclusion

We proposed an efficient strategy for inserting/deleting prototypes based on a certainty measure for online incremental prototype-based classification, in particular with metric adaptation, and we demonstrated its performance using GLVQ, GMLVQ, LGMLVQ in benchmarks. It turned out that the initialisation of the metric parameters only mildly effects the performance, while proper (i. e. small) learning rates are often crucial. Interestingly, the models are very robust to the choice of the initialisation strategies and parameters, whereby, as expected, the incremental parameters $g_{\max}$ and r_{num} control complexity and accuracy of the resulting models. Interestingly, most of the obtained values are Pareto optimal when varying these parameters. The obtained results are comparable or even better than the results obtained by batch learning and an incremental SVM version, displaying the great promise of the proposed models. One striking property of the results which we obtained in real life benchmarks is the comparably small

[3]Features are scaled to [0,1] for both SVMs.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

complexity with a number of prototypes which is larger than for batch variants but considerably smaller than the respective number of support vectors in SVM training. This observation identifies ioLVQ as a model with high promises for efficient online learning in the context of big data.

References

[1] P. Schneider, M. Biehl, and B. Hammer. Adaptive Relevance Matrices in Learning Vector Quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[2] S. Seo and K. Obermayer. Soft Learning Vector Quantization. *Neural Computation*, 15(7):1589–1604, Jul 2003.

[3] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[4] J. Read, A. Bifet, B. Pfahringer, and G. Holmes. Batch-Incremental versus Instance-Incremental Learning in Dynamic and Evolving Data. In *Proc. IDA*, pages 313–323, 2012.

[5] G. Cauwenberghs and T. Poggio. Incremental and Decremental Support Vector Machine Learning. In *Proc. NIPS*, pages 409–415, 2000.

[6] C.P. Diehl and G. Cauwenberghs. SVM Incremental Learning, Adaptation and Optimization. In *Proc. IJCNN*, volume 4, pages 2685–2690, 2003.

[7] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller. Incremental Support Vector Learning: Analysis, Implementation and Applications. *J. of Mach. Learning Res.*, 7:1909–1936, 2006.

[8] S. Bharitkar and D. Filev. An online learning vector quantization algorithm. In *Proc. of the 6th Int. Symp. on Signal Process. and its Applications*, pages 394–397, 2001.

[9] S. Kirstein, H. Wersing, and E. Körner. Rapid Online Learning of Objects in a Biologically Motivated Recognition Architecture. In *Proc. DAGM*, pages 301–308, 2005.

[10] S. Kirstein, H. Wersing, and E. Körner. A Biologically Motivated Visual Memory Architecture for Online Learning of Objects. *Neural Networks*, 21(1):65–77, 2008.

[11] T. C. Kietzmann, S. Lange, and M. Riedmiller. Incremental GRLVQ: Learning Relevant Features for 3D Object Recognition. *Neurocomputing*, 71(13-15):2868–2879, 2008.

[12] M. Grbovic and S. Vucetic. Learning Vector Quantization with Adaptive Prototype Addition and Removal. In *Proc. IJCNN*, pages 994–1001, 2009.

[13] Y. Xu, F. Shen, and J. Zhao. An incremental learning vector quantization algorithm for pattern classification. *Neural Computing and Applications*, 21(6):1205–1215, 2012.

[14] Bernd Fritzke. A Growing Neural Gas Network Learns Topologies. In *Advances in Neural Information Processing Systems, NIPS*, pages 625–632, 1994.

[15] L. Fischer, B. Hammer, and H. Wersing. Rejection Strategies for Learning Vector Quantization. In *Proc. ESANN*, pages 41–46, 2014.

[16] L. Fischer, B. Hammer, and H. Wersing. Efficient Rejection Strategies for Prototype-based Classification, 2014, Neurocomputing accepted.

[17] A. Sato and K. Yamada. Generalized Learning Vector Quantization. In *Advances in Neural Information Processing Systems*, volume 7, pages 423–429, 1995.

[18] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[19] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). *Technical Report CUCS-005-96*, February 1996.

[20] L. J. P. van der Maaten. Matlab Toolbox for Dimensionality Reduction, March 2013. http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html.

[21] Fred Henrik Hamker. Life-long learning Cell Structures–continuously learning without catastrophic interference. *Neural Networks*, 14(4-5):551–573, 2001.

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