ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

DropConnect for Evaluation of Classification Stability in Learning Vector Quantization

J. Ravichandran[1], S. Saralajew[2], and T. Villmann[1] *

1- Univ. of Appl. Sciences Mittweida,
Saxony Institute for Comp. Intelligence and Machine Learning
Computational Intelligence Research Group
Mittweida - Germany

2- Dr. Ing. h.c. F. Porsche AG - Driver Assistance Systems
Weissach - Germany

Abstract. In this paper we consider DropOut/DropConnect techniques known from deep neural networks to evaluate the stability of learning vector quantization classifiers (LVQ). For this purpose, we consider the LVQ as a multilayer network and transfer the respective concepts to LVQ. Particularly, we consider the output as a stochastic ensemble such that an information theoretic measure is obtained to judge the stability level.

1 Introduction

Dropout techniques like DropOut (DO) and DropConnect (DC) for deep multilayer perceptron networks (deep MLP, [1]) are appropriate methods to prevent the network from overfitting [2, 3]. DO can also be used during the working phase to judge the output's confidence level of the MLP [4]. Depending on task, the output of the MLP could either be a regression value or a class label.

Learning vector quantization (LVQ) is a prototype based classifier introduced by geometric considerations [5]. More specifically, LVQ distributes prototypes in the data feature space equipped with a dissimilarity measure depending on the class distribution given in this feature space. Nonetheless, LVQ can be also seen as a neural network as explained in [6]. Taking this perspective, dropout techniques can also be employed for LVQ. In this contribution we investigate a DC-approach for confidence estimation in LVQ networks to judge the classification stability. Particularly we focus on Generalized Matrix LVQ (GMLVQ) as it is one of the most powerful LVQ variants [7], which provides an interpretable sparse classifier with a performance comparable to that of deep networks for many problems [8, 9].

2 Generalized Learning Vector Quantization

We take the geometric perspective in considering GMLVQ. GMLVQ is a robust LVQ variant based on a cost function approximating the classification error [10, 7]. We assume data classes $1, \ldots, C$ and data $\mathbf{x} \in X \subseteq \mathbb{R}^n$ in the data space X. The aim is to distribute a set $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_M\} \subset \mathbb{R}^n$ of prototypes such that we can assign a class $c(\mathbf{x})$ to each $\mathbf{x} \in X$. Each prototype $\mathbf{w}_k$ is equipped with a class label $c(\mathbf{w}_j)$ such that at least one prototype is responsible for each class. Then the class assignment $c(\mathbf{x}) = c\left(\mathbf{w}_{s(\mathbf{x})}\right)$ for a data sample $\mathbf{x}$ is realized by means of a winner-take-all competition (WTAC)

$$s(\mathbf{x}) = \operatorname{argmin}_j \left(d(\mathbf{x}, \mathbf{w}_j)\right) \tag{1}$$

*J.R. is supported by a PhD-grant of ESF.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

where d is a given general dissimilarity measure [11]. We denote $\mathbf{w}_{s(\mathbf{x})}$ as the winner prototype of the competition. In standard LVQ, d is chosen as the squared Euclidean distance (SDE) whereas for the original GMLVQ the SDE

$$\delta_{\mathbf{\Omega}}(\mathbf{x}, \mathbf{w}_j) = (\mathbf{\Omega}(\mathbf{x} - \mathbf{w}_j))^2 \tag{2}$$

of the *projected data* with the projection matrix $\mathbf{\Omega} \in \mathbb{R}^{n_p \times n}$ is applied, and n_p is the projection dimension. The cost function to be minimized is $E_{GMLVQ}(W, \mathbf{\Omega}) = \sum_{\mathbf{x}} E(\mathbf{x}, W, \mathbf{\Omega})$ with local errors $E(\mathbf{x}, W, \mathbf{\Omega}) = \varphi(\mu(\mathbf{x}, W, \mathbf{\Omega}))$. Here $\varphi(z)$ is a monotonically increasing function frequently chosen as the identity function $\mathrm{id}(z) = z$ or the sigmoid function [12]. Further,

$$\mu(\mathbf{x}, W, \mathbf{\Omega}) = \frac{\delta_{\mathbf{\Omega}}(\mathbf{x}, \mathbf{w}^+) - \delta_{\mathbf{\Omega}}(\mathbf{x}, \mathbf{w}^-)}{\delta_{\mathbf{\Omega}}(\mathbf{x}, \mathbf{w}^+) + \delta_{\mathbf{\Omega}}(\mathbf{x}, \mathbf{w}^-)} \tag{3}$$

is the so-called classifier function where $\mathbf{w}^+$ is the best matching prototype with the correct class label $c(\mathbf{x}) = c\left(\mathbf{w}_{s(\mathbf{x})}\right)$ and $\mathbf{w}^-$ is the best matching prototype with an incorrect class label $c(\mathbf{x}) \neq c\left(\mathbf{w}_{s(\mathbf{x})}\right)$. Thus, $\mu(\mathbf{x}_k, W, \mathbf{\Omega}) \in [-1, 1]$ takes negative values if $\mathbf{x}$ is correctly classified.

Usually, learning in GMLVQ takes place as stochastic gradient descent learning (SGDL) for E_{GLVQ} according to

$$\Delta \mathbf{w}^{\pm} \propto -\xi\left(\mathbf{x}, \mathbf{w}^{\pm}\right) \cdot \frac{\partial \mu}{\partial \delta^{\pm}_{\mathbf{\Omega}}(\mathbf{x})} \frac{\partial \delta^{\pm}_{\mathbf{\Omega}}(\mathbf{x})}{\partial \mathbf{w}^{\pm}} \tag{4}$$

where the scaling factor

$$\xi\left(\mathbf{x}, \mathbf{w}^{\pm}\right) = \frac{\partial E(\mathbf{x})}{\partial \varphi} \cdot \frac{\partial \varphi}{\partial \mu} \tag{5}$$

is obtained by applying the chain rule for differentiation with the short hand notation $\delta^{\pm}_{\mathbf{\Omega}}(\mathbf{x}) = \delta_{\mathbf{\Omega}}(\mathbf{x}, \mathbf{w}^{\pm})$. The projection matrix $\mathbf{\Omega}$ also can be adjusted using SGDL by

$$\Delta \mathbf{\Omega} \propto -\xi\left(\mathbf{x}, \mathbf{w}^{\pm}\right) \cdot \frac{\partial \mu}{\partial \mathbf{\Omega}}$$

where

$$\frac{\partial \mu}{\partial \mathbf{\Omega}} = \frac{\partial \mu}{\partial \delta^{-}_{\mathbf{\Omega}}(\mathbf{x})} \cdot \frac{\partial \delta^{+}_{\mathbf{\Omega}}(\mathbf{x})}{\partial \mathbf{\Omega}} + \frac{\partial \mu}{\partial \delta^{-}_{\mathbf{\Omega}}(\mathbf{x})} \cdot \frac{\partial \delta^{-}_{\mathbf{\Omega}}(\mathbf{x})}{\partial \mathbf{\Omega}} \tag{6}$$

is the derivative of the classifier function with

$$\frac{\partial \mu}{\partial \delta^{+}_{\mathbf{\Omega}}(\mathbf{x})} = \frac{+2\delta^{-}_{\mathbf{\Omega}}(\mathbf{x})}{\left(\delta^{+}_{\mathbf{\Omega}}(\mathbf{x}) + \delta^{-}_{\mathbf{\Omega}}(\mathbf{x})\right)^2} \text{ and } \frac{\partial \mu}{\partial \delta^{-}_{\mathbf{\Omega}}(\mathbf{x})} = \frac{-2\delta^{+}_{\mathbf{\Omega}}(\mathbf{x})}{\left(\delta^{+}_{\mathbf{\Omega}}(\mathbf{x}) + \delta^{-}_{\mathbf{\Omega}}(\mathbf{x})\right)^2} \,. \tag{7}$$

As an alternative to explicit SGDL, advanced stochastic gradient approximations such as `AdaDelta`, `Adam` and `vSGDL` were investigated recently for use in GMLVQ [13]. The two latter algorithms performed best in this study and are hence highly recommended.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

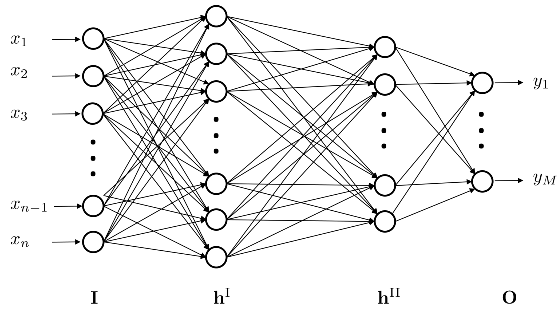

Fig. 1: Illustration of a LVQ-MLN with two hidden layers.

3 GMLVQ as a Multilayer Network

If we consider the squared distance

$$d_{\mathbf{\Omega}}(\mathbf{x}, \mathbf{w}_k) = (\mathbf{\Omega}\mathbf{x} - \mathbf{w}_k)^2 \tag{8}$$

for GMLVQ, the prototypes will no longer live in the data space but rather in the projection space, i.e. $\mathbf{w}_{\boldsymbol{k}} \in \mathbb{R}^{n_p}$. By doing so, we can consider GMLVQ as a multilayer network denoted as LVQ-MLN [6]: A LVQ-MLN consists of an input layer $\mathbf{I}$, two hidden layers $\mathbf{h}^{\mathrm{I}}$ and $\mathbf{h}^{\mathrm{II}}$ and an output layer $\mathbf{O}$, see Fig. 1. The nodes h_i^{I} of the first hidden layer $\mathbf{h}^{\mathrm{I}} \in \mathbb{R}^{n_p}$ are perceptron units according to

$$h_i^{\mathrm{I}}(\mathbf{x}) = g_i^{\mathrm{I}}\left(\langle \boldsymbol{\omega}_i, \mathbf{x}\rangle_E + \beta_i^{\mathrm{I}}\right) \tag{9}$$

with activation functions g_i^{I}, perceptron weight vectors $\boldsymbol{\omega}_i \in \mathbb{R}^n$ and biases $\beta_i^{\mathrm{I}} \in \mathbb{R}^n$. Thus, the first layer may perform a nonlinear projection

$$\mathbf{h}^{\mathrm{I}}(\mathbf{x}) = \mathbf{g}^{\mathrm{I}}_{\mathbf{\Omega},\boldsymbol{\beta}}(\mathbf{x}) \tag{10}$$

of the data depending on the choice of activation functions $\mathbf{g}^{\mathrm{I}}_{\mathbf{\Omega},\boldsymbol{\beta}}$ with $\mathbf{\Omega} = \left(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{n_p}\right)$ and the bias vector $\boldsymbol{\beta} \in \mathbb{R}^{n_p}$. Therefore, this layer is denoted as the projection layer in this context. The second layer $\mathbf{h}^{\mathrm{II}}$ is fully connected to the previous layer $\mathbf{h}^{\mathrm{I}}$ via

$$h_j^{\mathrm{II}}(\mathbf{x}) = g^{\mathrm{II}}\left(d\left(\mathbf{h}^{\mathrm{I}}(\mathbf{x}), \mathbf{w}_j\right)\right) \tag{11}$$

thereby realizing the prototype response. Here, d is an arbitrary (differentiable) dissimilarity measure and g^{II} is the activation function for the second layer which is usually chosen as the identity function $\mathrm{id}(z) = z$. For a crisp classifier network, the output layer $\mathbf{O} \in \mathbb{R}^M$ is calculated as

$$O_l = \sum_{k=1}^{M} H\left(h_l^{\mathrm{II}}(\mathbf{x}) - h_k^{\mathrm{II}}(\mathbf{x})\right) \tag{12}$$

where

$$H(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{else} \end{cases}$$

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

is the Heaviside function. Hence, O_l returns the winning rank of the prototype $\mathbf{w}_l$ and $O_l = 1$ is valid iff $l = s(\mathbf{x})$ with

$$s(\mathbf{x}) = \operatorname{argmin}_k \left(h_k^{\mathrm{II}}(\mathbf{x}) \right) \tag{13}$$

realizing the WTAC (1). Therefore, we denote the output layer also as the competition layer. Finally, the data point $\mathbf{x}$ is assigned to the class of the corresponding winning output unit $c\left(\mathbf{w}_{s(\mathbf{x})}\right)$. Thus, the formula (12) for the determination of the winning rank is equivalent that known from the neural gas network [14].

For a probabilistic or possibilistic LVQ, the output can be calculated as

$$O_l = \exp\left(\frac{-h_l^{\mathrm{II}}(\mathbf{x})}{\sum_k h_k^{\mathrm{II}}(\mathbf{x})} - \gamma \right) \tag{14}$$

realizing a softmax function. For each class $c \in \{1, \ldots, C\}$, the assignment probability $p_c(\mathbf{x})$ is calculated as

$$p_c(\mathbf{x}) = \sum_{k=1, c(\mathbf{w}_k)=c}^{M} O_k \tag{15}$$

as known from robust soft LVQ [15].

Obviously, if $\beta_i^{\mathrm{I}} = 0$ and $g_i^{\mathrm{I}}(z) = \mathrm{id}(z)$ is the identity for all $i = 1 \ldots n_p$, the projection in the projection layer simply becomes $\mathbf{h}^{\mathrm{I}}(\mathbf{x}) = \boldsymbol{\Omega}\mathbf{x}$, as it is required for (8) and, hence, the standard GMLVQ model is obtained. If a kernel distance $d_\kappa(\mathbf{x}, \mathbf{w}_j)$ with kernel $\kappa(\mathbf{x}, \mathbf{w}_j)$ is used as a dissimilarity measure in the prototype layer $\mathbf{h}^{\mathrm{II}}$, an implicit kernel mapping $\Phi_\kappa(\mathbf{x})$ takes place in the prototype layer [16].

As for standard GMLVQ, learning in LVQ-MLN can be realized by SGDL with respect to the prototypes and the projection matrix $\boldsymbol{\Omega}$ and the bias vector $\boldsymbol{\beta}$ as explained in [6].

4 DropConnect in LVQ-MLN and Classification Stability

The DC-concept can be easily realized in the LVQ-MLN applying it to the matrix $\boldsymbol{\Omega}$ of the projection layer $\mathbf{g}^{\mathrm{I}}_{\boldsymbol{\Omega},\boldsymbol{\beta}}(\mathbf{x})$ from (10). General DC simply sets values ω_{ij} to zero with a DC-probability p. Structured DC can be achieved if a certain component j^* is set to zero for all ω_{ij^*}. This is equivalent to DO of input units [6]. DC applied for a given data $\mathbf{x}$ in the working phase of a LVQ-MLN realizes a stochastic ensemble of C GMLVQ-output models $\mathcal{M}(\mathbf{x})$, which yield a probability distribution/density $P_{\mathcal{M}}(\mathbf{x}, c|\, p)$ regarding the classes c. A high network classification stability is given if only a few values are non-vanishing. Maximum instability is obtained if $P_{\mathcal{M}}(\mathbf{x}, c|\, p) \approx 1/C$ is resulted for all c. This behavior can be estimated using a information theoretic stability measure $\mathcal{C}(\mathbf{x}, c|\, p) = 1 - S(\mathbf{x}, c|\, p)/S_{\max}$ where

$$S(\mathbf{x}, c|\, p) = -\sum_c P_{\mathcal{M}}(\mathbf{x}, c|\, p) \cdot \log\left(P_{\mathcal{M}}(\mathbf{x}, c|\, p)\right) \tag{16}$$

is the Shannon entropy and $S_{\max} = \log(C)$ is its maximum value. Unique classification, i.e. maximum stability, is achieved for $\mathcal{C}(\mathbf{x}, c|\, p) = 1$. Another

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Type	n_p	p=0.5	p=0.1	p=0.01	accuracy
DrC	2	0.90 ± 0.05	0.94 ± 0.05	0.98 ± 0.04	100.00%
DrC	5	0.89 ± 0.05	0.94 ± 0.05	0.98 ± 0.03	100.00%
DrC	10	0.92 ± 0.05	0.96 ± 0.05	0.98 ± 0.04	100.00%
DiC	2	0.98 ± 0.04	0.99 ± 0.03	1.00 ± 0.01	100.00%
DiC	5	0.98 ± 0.03	1.00 ± 0.02	1.00 ± 0.01	100.00%
DiC	10	0.98 ± 0.03	0.99 ± 0.02	1.00 ± 0.01	100.00%

Table 1: Results of the stability analysis of a LVQ-MLN for the *Tecator* data set using different projection dimensions n_p for $\mathbf{\Omega}$. The last column reports the achieved classification accuracy for the undisturbed model. The p-value is the disturbance probability.

Type	n_p	p=0.5	p=0.1	p=0.01	accuracy
DrC	2	0.88 ± 0.10	0.85 ± 0.09	0.94 ± 0.05	89.86%
DrC	10	0.86 ± 0.11	0.93 ± 0.08	0.98 ± 0.04	91.58%
DiC	2	0.95 ± 0.06	0.98 ± 0.04	1.00 ± 0.02	89.86%
DiC	10	0.99 ± 0.04	0.99 ± 0.03	1.00 ± 0.01	91.58%

Table 2: Results of the stability analysis of a LVQ-MLN for the *FLC* dataset. The interpretation is as for Tab. 1.

option for stability estimation is to add random noise to the weights, i.e. $\hat{\omega}_{ij} = \omega_{ij} + \eta$ of small variance relative to $|\omega_{ij}|$.

5 Application and experimental results

We tested the approach for two real world data sets. The first one is the well-known *Tecator* [17] of spectral data to detect the fat content in meat (binary classification). The second data set (*FLC*) is a classification data set for LANDSAT TM data vectors in ground cover classification (11 classes) [18]. The results of the experiments that were performed on the *Tecator* data set are summarized in the Tab. 1. The values in the cells are the averaged stability values (± standard deviation) $\mathcal{C}(\mathbf{x}, c|p)$ of the data points. In all experiments only one prototype per class was used. The disturbance level for DiC was set to $\eta = 1\%$.

The results of the experiments that were performed on the *FLC* are summarized in the Tab. 2 with the same interpretation as for *Tecator*.

For both experiments we can conclude that the suggested procedure provides an appropriate method to evaluate the classification stability.

6 Conclusion

In this contribution we considered the problem of evaluation of classification certainty/stability in LVQ-MLN. It was tackled adopting the idea of DropConnect as known from deep networks. Thus a measure of confidence/stability for the classification can be estimated during the working phase of the network. Future work will include the investigation of reject options, as explained in [19], into this approach to further improve the classifier stability as well as the evaluation of class-dependent stability.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[3] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML), Atlanta, Georgia, USA*, volume 28 of *JMLR:W&CP*, pages 1–9, 2013.

[4] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In M.F. Balcan and K.Q. Weinberger, editors, *Proceedings of the International Conference on Machine Learning, New York, New York, USA*, volume 48, pages 1050–1059, 2016.

[5] Teuvo Kohonen. Learning Vector Quantization. *Neural Networks*, 1(Supplement 1):303, 1988.

[6] T. Villmann, S. Saralajew, A. Villmann, and M. Kaden. Learning vector quantization methods for interpretable classification learning and multilayer networks. In C. Sabourin, J.J. Merelo, A.L. Barranco, K. Madani, and K. Warwick, editors, *Proceedings of the 10th International Joint Conference on Computational Intelligence (IJCCI), Sevilla*, pages 15–21, Lissabon, Portugal, 2018. SCITEPRESS - Science and Technology Publications, Lda. ISBN: 978-989-758-327-8?

[7] P. Schneider, B. Hammer, and M. Biehl. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21:3532–3561, 2009.

[8] M. Biehl, B. Hammer, and T. Villmann. Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 7(2):92–111, 2016.

[9] T. Villmann, A. Bohnsack, and M. Kaden. Can learning vector quantization be an alternative to SVM and deep learning? *Journal of Artificial Intelligence and Soft Computing Research*, 7(1):65–81, 2017.

[10] A. Sato and K. Yamada. Generalized learning vector quantization. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, pages 423–9. MIT Press, Cambridge, MA, USA, 1996.

[11] D. Nebel, M. Kaden, A. Villmann, and T. Villmann. Types of (dis−)similarities and adaptive mixtures thereof for improved classification learning. *Neurocomputing*, 268:42–54, 2017.

[12] T. Villmann, J. Ravichandran, A. Villmann, D. Nebel, and M. Kaden. Activation functions for generalized learning vector quantization - a performance comparison. Technical Report arXiv:1901.05995, ArXiv, 2019.

[13] M. LeKander, M. Biehl, and H. deVries. Empirical evaluation of gradient methods for matrix learning vector quantization. In *Proceedings of the 12th Workshop on Self-Organizing Maps and Learning Vector Quantization (WSOM2017+)*, pages 1–8. IEEE Press, 2017.

[14] Thomas M. Martinetz, Stanislav G. Berkovich, and Klaus J. Schulten. 'Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Trans. on Neural Networks*, 4(4):558–569, 1993.

[15] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589–1604, 2003.

[16] T. Villmann, S. Haase, and M. Kaden. Kernelized vector quantization in gradient-descent learning. *Neurocomputing*, 147:83–95, 2015.

[17] F. Rossi, N. Delannay, B. Conan-Gueza, and M. Verleysen. Representation of functional data in neural networks. *Neurocomputing*, 64:183–210, 2005.

[18] D.A. Landgrebe. *Signal Theory Methods in Multispectral Remote Sensing*. Wiley, Hoboken, New Jersey, 2003.

[19] T. Villmann, M. Kaden, A. Bohnsack, S. Saralajew, J.-M. Villmann, T. Drogies, and B. Hammer. Self-adjusting reject options in prototype based classification. In E. Merényi, M.J. Mendenhall, and P. O'Driscoll, editors, *Advances in Self-Organizing Maps and Learning Vector Quantization: Proceedings of 11th International Workshop WSOM 2016*, volume 428 of *Advances in Intelligent Systems and Computing*, pages 269–279, Berlin-Heidelberg, 2016. Springer.