ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

RecLVQ: Recurrent Learning Vector Quantization

Jensun Ravichandran , Marika Kaden , and Thomas Villmann *

University of Applied Sciences Mittweida,
Saxon Institute for Comp. Intelligence and Machine Learning
Mittweida - Germany

Abstract. Learning Vector Quantizers (LVQ) and its cost-function-based variant called Generalized Learning Vector Quanitzation (GLVQ) are powerful, yet simple and interpretable classification models. Even though GLVQ is an effective tool for classifying vectorial data, it cannot handle raw sequence data of potentially different lengths. Usually, this problem is solved by manually engineering fixed-length features or by employing recurrent networks. Therefore, a natural idea is to incorporate recurrent units for data processing into the GLVQ network structure. The processed data can then be compared in a latent space for classification decisions. We demonstrate the ability of this approach on illustrative classification problems.

1 Introduction

GLVQ is a prototype-based sparse classification model that can be trained using gradient descent schemes. It constitutes a robust classifier based on data dissimilarities optimizing the hypothesis margin [1]. On the other hand, it is well known that recurrent architectures like the Long Short Term Memory (LSTM) network [2] and Gated Recurrent Unit (GRU) [3] are powerful models to process sequential data. However, so far, GLVQ-type networks have not been combined with recurrent networks even though there are obvious advantages in doing so for sequence processing. RecLVQ provides a framework for incorporating deep-recurrent neural networks into GLVQ to take advantage of some desirable properties including good robustness and margin maximization [1, 4]. Vector quantization utilizing a recursive structure has so far only been investigated in unsupervised learning by applying Self-Organizing Maps (SOM) [5, 6] for sequence processing where the SOM-structure is used to manage the recursive data structure in sequences.

2 Related Work

Recently, it was proposed to incorporate *prototype layers* into deep neural network architectures [7]. Previously, however, the research communities of vector quantization and neural networks were perceived to be working on models that were incompatible with each other. Bridging the gap between the communities allows for the crossover of certain ideas between the two fields. An example for such applied crossover is the use of Dropout [8] and DropConnect [9] for stability estimation in the context of LVQ [10]. We strongly believe that the two fields

*M.K. and J.R. are supported by grants of the European Social Fund (ESF).

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

have much to gain by adapting methods and techniques from each other in the near future.

Outside the LVQ community, there has been recent interest in similar ideas to incorporate prototype layers in recurrent networks. One such attempt is ProSeNet [11]. The main difference of our architecture as compared with ProSeNet is the use of a Siamese structure. Yet another comparable network structure, albeit with a different loss function than GLVQ (explained in 3.1) was introduced in [12]. However, because the prototypes are in the latent space, they propose to additionally learn an inverse mapping that reproduces the prototypes in the original space. We find this cumbersome and unnecessary, as the same can be achieved quite elegantly by means of a Siamese architecture. Having said that, learning such an inverse mapping might be useful for "generating" new data samples by sampling from a low dimensional embedding space.

3 Modern Variants of Learning Vector Quantization

3.1 Generalized Learning Vector Quantization

In 1996, GLVQ [13] was proposed by introducing a loss function which was a soft (differentiable) approximation to the misclassification error. In GLVQ, it is assumed that the training set $\mathcal{X}_{train} = \{(\boldsymbol{x}_k, y_k)\}$ consists of tuples of data vectors $\boldsymbol{x_k} \in \mathbb{R}^{n_x}$ and their respective class labels $y_k \in \mathcal{C} = \{c_k\}_{k=1}^{|\mathcal{C}|}$. A similar set $\mathcal{W} = \{(\boldsymbol{w}_k, \lambda_k)\}_{k=1}^{|\mathcal{W}|}$ called the prototype set, consisting of trainable prototype vectors $\boldsymbol{w}_k \in \mathbb{R}^{n_x}$ and their respective class labels λ_k such that all classes in the set $\mathcal{C}$ are represented, is also assumed. For the sake of convenience, we will introduce a unified labeling function c which outputs the true class labels for training data, i.e., $c(\boldsymbol{x_k}) = y_k$ and prototype labels in the case prototypes, i.e., $c(\boldsymbol{w_k}) = \lambda_k$. The goal of GLVQ is then to distribute the prototype vectors such that the class label of any new input $\boldsymbol{x}$ can be inferred by means of a Winner-Takes-All Competition (WTAC) given by $c(\boldsymbol{w}_{s(\boldsymbol{x})})$ where $s(\boldsymbol{x}) = \text{argmin}_k\, d(\boldsymbol{x}, \boldsymbol{w}_k)$. This is achieved by minimizing the GLVQ loss function as given by:

$$\mathcal{L} = \sum_{\mathcal{X}_{train}} f\left(\frac{d(\boldsymbol{x}, \boldsymbol{w}^+) - d(\boldsymbol{x}, \boldsymbol{w}^-)}{d(\boldsymbol{x}, \boldsymbol{w}^+) + d(\boldsymbol{x}, \boldsymbol{w}^-)}\right) \tag{1}$$

where $\boldsymbol{w}^+$ is the closest prototype to $\boldsymbol{x}$ with a matching label such that $c(\boldsymbol{x}) = c(\boldsymbol{w}^+)$, $\boldsymbol{w}^-$ is the closest prototype to $\boldsymbol{x}$ with a non-matching label such that $c(\boldsymbol{x}) \neq c(\boldsymbol{w}^+)$ and f is usually a monotonically increasing transfer function.

3.2 Generalized Matrix Leaning Vector Quantization (GMLVQ)

3.2.1 The Canonical Perspective of GMLVQ

One of the drawbacks of LVQ variants that use an unparameterized dissimilarity measure like the Euclidean distance for example is that they weigh all input dimensions equally, which is an undesirable property for most practical applications. One of the first attempts to address this problem was proposed in [14] and the resulting LVQ scheme was called Generalized Relevance Learning Vector Quantization (GRLVQ). In GRLVQ, the distance measure itself is

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

adaptable along with the prototype vectors by means of gradient descent and a so-called *relevance profile* describes the relevance (weightage) of each dimension of the respective vector space where the input vectors and prototypes reside. Soon however, it was realized that the relevance profile could be generalized into a *relevance matrix*, and the resulting scheme was termed Generalized Matrix Learning Vector Quantization (GMLVQ) as the distance measure now included a full matrix of adaptable weights [15]. Here, the distance measure $d(\boldsymbol{x}, \boldsymbol{w})$ is changed to be an adaptive distance measure of the form $d_{\boldsymbol{\Omega}}(\boldsymbol{x}, \boldsymbol{w}) = (\boldsymbol{\Omega}(\boldsymbol{x}-\boldsymbol{w}))^2$, where $\boldsymbol{\Omega} \in \mathbb{R}^{n_m \times n_x}$ is a matrix of adaptable parameters with the desired mapping dimension n_m.

3.2.2 The Siamese Perspective of GMLVQ

Although the canonical GMLVQ scheme was contrived with the motivation of generalizing on the idea of a relevance profile from GRLVQ, it turns out that GMLVQ can be viewed quite differently. One such perspective, which we shall call the Siamese perspective of GMLVQ is obtained by rewriting the distance measure as $d_{\boldsymbol{\Omega}}(\boldsymbol{x}, \boldsymbol{w}) = (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{w}})^2$, where $\tilde{\boldsymbol{x}} = \boldsymbol{\Omega}\boldsymbol{x}$ and $\tilde{\boldsymbol{w}} = \boldsymbol{\Omega}\boldsymbol{w}$ is a linear mapping of the data and the prototypes. Even though doing so is mathematically trivial, it offers a vastly different perspective on closer inspection. Taking the Siamese perspective, the distance measure becomes unparameterized with the $\boldsymbol{\Omega}$ matrix acting as a linear transformation of the data and the prototypes, before being presented to the distance metric for measurement (see Fig. 1). Also notice that the matrix $\boldsymbol{\Omega}$ shares the trainable parameters even though it is used twice in a single forward-pass through the network; once to embed the input $\boldsymbol{x}$ and once to embed a prototype $\boldsymbol{w}$. It turns out that this structure is known as a "Siamese" network, albeit with one of the inputs being adaptable prototypes in the case of GMLVQ.

The earliest mention of a Siamese network appeared in 1993 [16]. The original motivation there was to transform two inputs into appropriate representations before comparing them. Since then, numerous papers have appeared that use the idea in many different ways. One of the inputs in such networks is usually a pre-chosen *anchor*, against which the true input is compared. However, in such cases, the errors are not back-propagated to the anchor to adapt it. The anchor, once chosen, is kept fixed throughout the training procedure. There are indeed two ways to make a network produce an output closer to the ground truth for a given input: adapt the network weights, or adapt the input to the network itself, by back-propagating the errors. We usually do not adapt the inputs though, because they are given and fixed, except perhaps in cases such as transfer learning or in finding an adversarial example.

3.3 RecLVQ

Recurrent architectures are powerful tools to process sequential data. In this section, we provide the RecLVQ framework for incorporating recurrent structures in GLVQ-type networks, starting with GMLVQ, taking again a Siamese perspective. RecLVQ replaces the linear mapping block with a recurrent processing unit taking a sequence input $\boldsymbol{x}$ and outputting $Rec(\boldsymbol{x})$. One choice for the Rec block could be an LSTM network (see Fig. 1). Obviously, a recurrent model

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

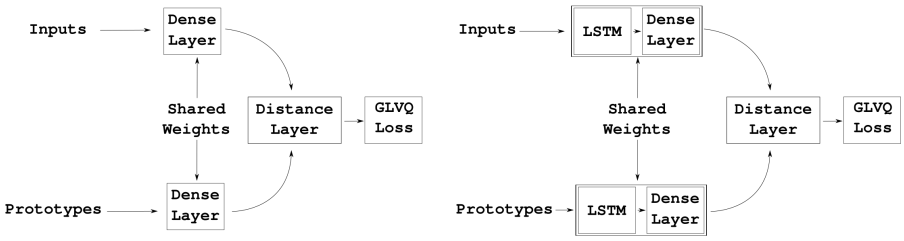

Fig. 1: On the left is the Siamese perspective of GMLVQ showing the shared mapping. On the right is a Siamese RecLVQ Network showing the connections between the constituent layers, including the shared recurrent mapping. Here the recurrent structure is chosen to be an LSTM layer for illustration.

such as LSTM has shared parameters across time steps in the input sequence. However, in the Siamese setting, these parameters are additionally shared across the two input branches of the network. With this, it now becomes plausible to map input sequences and prototypes of various sequence lengths to a fixed latent representation of chosen length before determining distances for the WTAC. To this end, the latent representation could be chosen as the hidden state of the LSTM cell from the last time step, for example. Another possibility is to use the concatenated cell outputs from the last timestep(s). We emphasize that the network illustrated in Fig. 1 is but one of many possible network configurations that the framework allows. Bidirectional LSTM networks, deep LSTM networks and combinations there of, are all plausible within the RecLVQ framework. The essential pieces of RecLVQ are a recurrent structure such as LSTM, a differentiable distance measure and the GLVQ loss function from Eq. (1) connected together in a Siamese fashion with shared weights for recurrent mapping. As long as these essential ingredients are present, it is possible to mix and match other differentiable functions into the network configuration as needed. One can think about the LSTM-unit as a kind of data processing to embed the sequence into an embedding space. In this embedding/latent space, GLVQ works as usual with guaranteed robustness [17, 18]. Moreover, the interpretability of GLVQ is kept partially, at least.

4 Experiments

We trained a RecLVQ mode with a deep bidirectional LSTM backbone, on the MNIST dataset to 98.5% accuracy on the test benchmark using only one prototype per class. We chose this dataset not necessarily because we believe that the rows of the images in the MNIST dataset are sequentially generated (although it is not a completely unreasonable assumption given that the digits are hand-drawn), but because it allows us to demonstrate the model on a sufficiently large dataset whilst still being able to visualize the high-dimensional prototypes

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

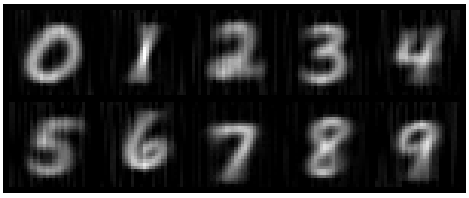

Fig. 2: RecLVQ prototypes after training on the MNIST dataset.

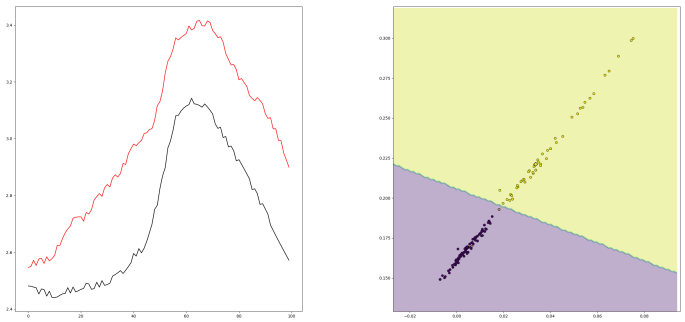

Fig. 3: On the left are the Tecator prototypes visualized as spectra after training and on the right is a visualization of the corresponding 2D-latent space of the Tecator data and the prototypes.

as images on paper. Another reason for this choice is that interpretability is a domain-specific notion that requires domain-expertise to be able interpret the prototypes using case-based reasoning. Our assumption here being that the readers are experts at recognizing hand-drawn Arabic-numerals.

We also trained a RecLVQ model on the Tecator dataset [19] to 100% on the test benchmark. Each data sample here consists of a 100 channel absorbance-spectrum measured from meat samples and the task is to categorize the data samples as either "High-Fat" or "Low-Fat". The learned-prototypes (as spectra) and the two-dimensional latent embeddings are both visualized in Fig. 3. Interpreting the prototypes here however, is unfortunately not as straight-forward and we shall not attempt to do so.

5 Conclusion

In this paper, we have proposed the use of a Siamese network structure to recurrently process sequential data of potentially varying lengths for classification tasks using prototype-modelling along with an explicit WTAC competition. In the experiments, we empirically show that it is plausible to learn prototypes for the different classes by backpropagating the errors through recurrent structure

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

in the model. As the next step, we intend to apply our proposed method to study DNA/RNA sequences. It is worth noting that models like GMLVQ are fully interpretable in that all layers in those networks lend themselves naturally to human interpretation. In RecLVQ however, the recurrent structure still remains a black-box. Eventhough RecLVQ is not fully interpretable in the same fashion as GLVQ or GMLVQ, it may be a reasonable price to pay for certain applications.

References

[1] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin analysis of the LVQ algorithm. In *NIPS*, 2002.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9, 1997.

[3] D. Bahdanau, K. H. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[4] S. Saralajew, L. Holdijk, and T. Villmann. Fast adversarial robustness certification of nearest prototype classifiers for arbitrary seminorms. In *NeurIPS*, 2020.

[5] T. Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15:979–991, 2002.

[6] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. A general framework for unsupervised processing of structured data. *Neurocomputing*, 57:3–35, 2004.

[7] S. Saralajew, L. Holdijk, M. Rees, and T. Villmann. Prototype-based neural network layers: incorporating vector quantization. *arXiv preprint arXiv:1812.01214*, 2018.

[8] N. Srivastava, G. E. 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.

[9] L. Wan, M. D. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1058–1066, 2013.

[10] J. Ravichandran, M. Kaden, S. Saralajew, and T. Villmann. Variants of dropconnect in learning vector quantization networks for evaluation of classification stability. *Neurocomputing*, 403:121–132, 2020.

[11] Y. Ming, P. Xu, H. Qu, and L. Ren. Interpretable and steerable sequence learning via prototypes. In *KDD*, pages 903–913, 2019.

[12] O. Li, H. Liu, C. Chen, and C. Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*, pages 3530–3537, 2018.

[13] A. Sato and K. Yamada. Generalized learning vector quantization. In *NIPS*, pages 423–429, 1995.

[14] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

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

[16] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a siamese time delay neural network. In *NIPS*, pages 737–744, 1993.

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

[18] S. Saralajew, L. Holdijk, M. Rees, and T. Villmann. Robustness of generalized learning vector quantization models against adversarial attacks. In *Advances in Intelligent Systems and Computing*, volume 976, 2020.

[19] Tecator dataset. `http://lib.stat.cmu.edu/datasets/tecator`, 1995. Contained in StatLib Datasets Archive.