Dynamic Prototype Addition in Generalized Learning Vector Quantization

Jonathon Climer and Michael J. Mendenhall

Abstract Learning Vector Quantization (LVQ) is a powerful supervised learning method for classification that uses a network of prototype vectors to form a decision surface. Generalization theory shows there is a non-trivial number of prototype vectors that yield the best generalization. Although it is typical to assign the same number of prototype vectors for each class, other LVQ methods add prototypes dynamically (incrementally) during training. This work offers an extension to the existing dynamic LVQs that minimizes the cost function of Generalized LVQ by focusing on the set of misclassified samples. This cost minimization occurs between the largest cost-contributing class and its nearest "confuser class". A comparison is made between other prototype insertion methods and compares their classification performance, the number of prototype resources required to obtain that accuracy, and the impact on the cost function.

Keywords Dynamic/incremental learning vector quantization · Large margin classifier · Cost minimization

1 Introduction

The family of Learning Vector Quantization (LVQ) [1] methods are supervised learners for statistical pattern recognition. They belong to a class of simple competitive learners and have gained popularity due to their efficiency, ease of implementation, and clear interpretability during training and classification. These algorithms are capable of classifying very high dimensional data and are applied in a variety of fields including machine vision [2, 3], analysis of medical imagery [4], and the classification of hyperspectral data [5].

J. Climer (✉) · M.J. Mendenhall
Department of Electrical and Computer Engineering, Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, 45433, USA
e-mail: jonathon.climer@afit.edu; jonclimer@gmail.com

E. Merényi et al. (eds.), *Advances in Self-Organizing Maps and Learning Vector Quantization*, Advances in Intelligent Systems and Computing 428,
DOI 10.1007/978-3-319-28518-4_31

LVQs use representative “prototype” or “code-book” vectors whose positions in the data space are updated during rounds of “learning”. Multiple prototypes per class are typically used in order to achieve accurate classifications. However, Crammer et. al. [6] shows that the generalization error of LVQ-based classifiers (that lead to a “winner-takes-all” classification) are a function of the number of prototype vectors used. Since the goal of any classifier is to generalize the decision surface, it follows that too many prototypes can lead to over-fitting and that there is a non-trivial number of prototypes among the classes for a given problem.

Solutions that do not make *a-priori* assumptions on the prototype distribution between classes add them *dynamically* or *incrementally* as part of the learning process (hereinafter *dynamic*). This concept is supported by [2, 3, 7, 8], based on LVQ2, LVQ3, GRLVQ, and GLVQ respectively. This paper considers a prototype insertion strategy to Generalized LVQ (GLVQ) that minimizes the cost function directly. Our method, in some cases, shows faster convergence due to larger accuracy gains early in the training process, and in some cases requires fewer prototype vectors than other methods in the same class [2, 3, 7, 8].

2 Learning Vector Quantization (LVQ) Background

2.1 LVQ Taxonomy

The taxonomy of LVQ is represented as three phases: competition, winner selection, and synaptic adaptation. Each LVQ defines the set of prototype vectors it allows to **compete**, commonly selecting one or more from the set of “in-class” prototypes (belonging to the same class as the input sample x), “out-of-class” prototypes (belonging to any class other than that of the input sample x), or a “net” prototype (chosen from all prototype vectors, regardless of class label). A **winning** prototype is one that results in the minimum distortion between it and the current sample. When using Euclidean distance as the distortion measure, the winner w_i represents the prototype from the set of competitors, closest to x. (Frequently in LVQ, a second competition selects one more winning prototype, w_j.) After winner selection, many LVQs impose additional conditions in order to apply updates to the prototypes based upon the influence of x, such as windowing functions [9]. Where the additional conditions are satisfied, the authors in [10] show the **synaptic adaptation** rule for LVQ algorithms can be generalized as:

$$w_i \leftarrow w_i - \alpha\frac{\partial S}{\partial w_i}; \qquad w_j \leftarrow w_j - \alpha\frac{\partial S}{\partial w_j}, \tag{1}$$

where α is the (potentially time varying) learn rate and S is the cost function. The completion (or termination) of these three phases for a sample x constitutes one training step. This process is then repeated for all training samples $\{x_1, x_2, \ldots, x_N\}$ to constitute one epoch (where N is the total number of training samples).

2.2 Dynamic LVQ Ancestry

After the introduction of LVQ in [1], several variants have arisen to better estimate the decision surface and overcome divergence and long-term stability challenges [9]. Poirier [7] introduced a new method based on Kohonen's LVQ2 to *dynamically* add prototype vectors as needed to better represent the class distributions and form better decision surfaces. Later works built dynamic LVQ (DLVQ) methods based on LVQ2.1 [11], LVQ3 [8, 12], GLVQ [2], and Relevance GLVQ (GRLVQ) [13]. Of these, the GLVQ variants appear to offer the strongest performance due to their use of a cost function that guides synaptic adaptation. Consequently, GLVQ forms the basis of the methods described in this paper. Following the taxonomy in Sect. 2.1, the competition and winner selection phase of GLVQ uses both the nearest in-class and nearest out-of-class prototypes (w_i and w_j respectively). Instead of explicitly restricting winner selection by a window about the midpoint between w_i and w_j, GLVQ implicitly does this [5, 10] by employing the cost [10]:

$$S = \sum_{n=1}^{N} f\left(\mu\left(x_n\right)\right), \mu\left(x\right) = \frac{d_j - d_i}{d_j + d_i}, \tag{2}$$

where N is the number of training samples and d_i and d_j are squared Euclidean distances between the input sample x_n and the prototype vectors w_i and w_j, respectively. Consistent with [10], $f\left(\mu\right)$ is is the sigmoid function $1/\left(1+e^{-\mu}\right)$. When $\partial S/\partial w_i$ and $\partial S/\partial w_j$ are substituted into Eq. (1), the resulting synaptic adaptation equations minimize the cost function via gradient descent [10]:

$$w_i \leftarrow w_i - \alpha\frac{\partial f}{\partial \mu}\frac{d_j}{\left(d_j + d_i\right)^2}\left[x - w_i\right]; \quad w_j \leftarrow w_j + \alpha\frac{\partial f}{\partial \mu}\frac{d_i}{\left(d_j + d_i\right)^2}\left[x - w_j\right]. \tag{3}$$

2.3 LVQ Taxonomy Addition: Network Structure Modification

In order to characterize the addition and/or deletion of prototype vectors in DLVQs, a new element is incorporated into the standard LVQ taxonomy. **Network structure modification** (NSM) captures the second dynamic and adaptive component that distinguishes DLVQs from its ancestors. This part of the taxonomy is responsible for identifying which class receives the new prototype and its initial location in the data space. While this paper does not exhaustively explore all DLVQ NSM methods in the literature, several are used in our comparative analysis.

Mean: In Zell et al. [14], new prototypes are initialized as the mean of the misclassified samples and is done so across all classes. Although [14] adds, a prototype to each class, we insert a single prototype in the class with the largest classification error.

Closest: Kirsten et al. [3] insert new prototypes into classes where the classification rate exceeds a threshold. This enables the addition of several prototypes at once. New prototypes are placed near class boundaries by initializing them at the same position as the misclassified sample closest to an out-of-class prototype vector. In our use of this method, we insert a single prototype in order to normalize the comparison between insertion methods.

Sampling Cost: Losing et al. [2] represents the latest in DLVQs and has the same overarching goal described in this paper. That is, they desire to minimize the cost function directly in order to maximize the classification accuracy. Conceptually, this is achieved by selecting a random subset of the training samples (e.g., a fixed percentage) as candidate positions for the insertion of a new prototype. One-by-one, candidate positions are tested by calculating the total cost among the subset after the candidate has been added. The position (and class label) of the candidate resulting in the lowest total cost is chosen as the initialization of the new prototype.

Principal Components (PC): Stefano et al. [8] tracks the number of times in-class and out-of-class prototypes are referenced in order to calculate a *split metric* for each prototype. Prototypes with split metrics exceeding a threshold are *split* by replacing the current prototype with two new prototypes placed equidistant from the original along the principal component (eigenvector) direction of the target class's misclassified samples. The distance along the principal component axis is a function of the associated variance (eigenvalue for the corresponding eigenvector).

2.4 Two Proposed NSM Methods

Our *Near-Mean* method is similar to the *Mean* method in [14], but restricts the new prototype to the misclassified sample with the smallest Euclidean Distance to the mean of the misclassified samples. By assuming a "known valid" position closest to the average, we reduce the potential of poorly interpolating the initialization of the new prototype in a sparse area that may represent irregularities in the class distribution, or even represent another class.

Our *Misclassified Cost* method seeks to minimize the cost function directly by focusing on misclassified samples. Conceptually, our method accumulates the cost contribution from each misclassified sample according to Eq. 2 in a confusion matrix according to "true class" and "nearest class". The two classes whose interaction results in the highest total cost define the pool of candidate locations for the new prototype: specifically, their misclassified samples. Each of the candidate solutions are tested and the one with the lowest cost per Eq. 2 is chosen.

2.5 Qualitative Comparison of NSM Methods

The presented NSM methods vary in how they try to improve the configuration of the LVQ network in hopes of improving classification accuracy and generalization. The *Misclassified Cost*, *Mean*, *Near-Mean*, and *Closest* methods utilize the set of misclassified training samples to select a recipient class for a new prototype, restricting prototype placement to the class with largest classification error. The *Sampling Cost* and *PC* methods select a candidate location globally (across all samples) where *Sampling Cost* selects from a random subset of the training data. *PC* relies upon misclassified data from each training sample to compute its split metric and assess prototype utilization.

The methods also vary in candidate locations for the new prototype and the resource burden associated with it. *Sampling Cost* and *Misclassified Cost* both perform direct minimization of the cost function. *Sampling Cost* compares placement at $\tilde{N}$ potential locations calculating the cost over all $\tilde{N}$ training samples (in our implementation, $\tilde{N} = N/10$). *Misclassified Cost* calculates the cost over all misclassified samples and once the two classes from the most expensive two-class boundary are identified, candidate locations are evaluated only over the misclassified samples for those two classes. *Misclassified Cost* down selects candidate positions in a way that offers a reduction in total operations. As classification accuracy improves, NSM using the misclassified samples as the candidate locations will reduce in computational complexity as the pool (typically) reduces over continued training. This is in contrast to *Sampling Cost* where the number of samples evaluated as candidate locations remains fixed.

In both cost motivated methods, as well as *Closest* and *Near-Mean*, new prototypes are initialized in positions of known training samples. However, *Mean* and *PC* allow the initialized position to be anywhere. This less restrictive initialization may be beneficial, however, it may also lead to formation of prototypes in sparse (or poorly defined) regions of the *pdf*. Additionally, the necessary computations can be complex. In the case of *Mean*, it is simply the average location of the misclassified samples within the class. *PC* however, requires the additional computation of the covariance matrix and its eigenvectors and eigenvalues.

3 Experimental Process and Results

The NSM methods are compared within the framework of a dynamic GLVQ over three data sets: the Mice Protein Expression data set [15], the USPS Handwritten image data set [16], and the Lunar Crater Volcanic Field hyperspectral data set [17]. In order to promote a fair comparison of NSM methods and allow networks to converge, we restrict the potential for NSM to occur after a fixed number of epochs, allowing at most one new prototype per fixed interval. Consequently, the *Closest* and *Mean* methods add a prototype (when appropriate) to the class with the largest classification error. Training and testing partitions for each data set are preserved between NSM methods, and prototypes in each are initialized to the same values. Additionally, the number of samples used in *Sampling Cost* is restricted to 10 % of the training set, selected randomly.

3.1 Data Sets and Experiment Setup

The Mice Protein Expression data set [15] reports protein expression levels in the brains of eight classes of mice with varying biology and treatments. While the report sought to identify individual proteins linked to learning and Down Syndrome [18], it presents an interesting classification problem. Due to empty fields in the data, 71 of 77 features are selected for 1047 samples. K-Fold cross-validation with $K = 5$ [19] is used to train and validate classification performance. For consistency in training, the protein expression features are linearly scaled on [0, 1]. A learn rate of $\alpha = 0.005$ is used and prototype insertion occurs after 100 epochs.

The USPS Handwritten data set [16] has 9,298 samples where each sample is a 16×16 pixel scan of a handwritten digit $\{0, 1, \ldots, 9\}$. Each 16×16 scan is "linearized" creating a vector with 256 features and each feature is linearly scaled on [0, 1]. K-fold cross-validation is used with $K = 5$ to train and validate performance. An $\alpha = 0.1$ is used and prototype insertion occurs after 150 epochs.

The Lunar Crater Volcanic Field (LCVF) hyperspectral data set [17] contains 1464 samples drawn from 35 classes, each with 194 spectral dimensions. As recommended in [20], each feature vector is normalized with its ℓ_2-norm to compensate for the effect of shadowing due to sensor geometry. Due to the sparsity of several classes, K-fold cross-validation is used with $K = 3$. The learn rate is $\alpha = 0.00001$ and prototype insertion occurs after 100 epochs.

3.2 Results and Discussion

This section introduces the graphs and tables used to draw specific discussion on each data set in Sects. 3.3, 3.4, and 3.5 for the Mice, USPS, and LCVF data sets respectively. The classification and cost-minimization performance on the three data sets previously described is shown in Figs. 1 and 2. They show the performance of *PC*, *Sampling Cost*, *Misclassified Cost*, *Mean*, and *Closest*. In order to improve readability of the figures, *Near-Mean* is omitted. For a baseline, GLVQ as described in [10] is used and is initialized with the maximum number of prototypes listed on the plots. Table 1 shows the training and number of prototypes required to exceed baseline GLVQ accuracy. Locations with a '–' identify NSM methods that did not meet the baseline performance. The strongest overall performers with the peak classification accuracy of each NSM method is listed in Table 2. These numbers are reported along with the total cost and required number of prototypes for the reported configurations.

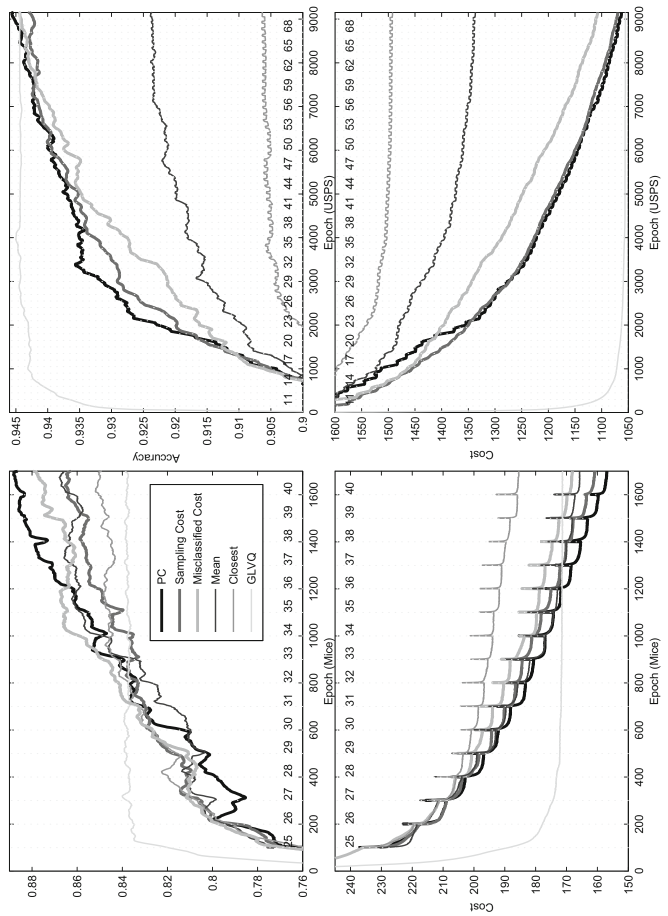

Fig. 1 Accuracy (*top*) and cost function value (*bottom*) versus testing epoch for DGLVQ with the following NSM methods: *PC*, *Sampling Cost*, *Misclassified Cost*, *Mean*, *Closest*, and GLVQ for the Mice (*left*) and USPS (*right*) data sets. GLVQ is evaluated using the maximum number of allocated prototypes and is included for reference. Plots are the average of the $K = 5$ validation results

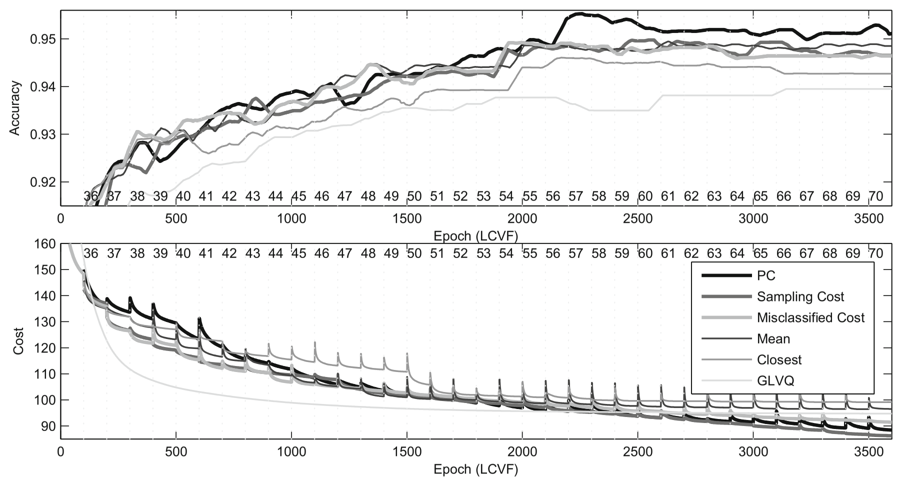

Fig. 2 Accuracy (*top*) and cost function value (*bottom*) versus testing epoch for DGLVQ with the following NSM methods: *PC*, *Sampling Cost*, *Misclassified Cost*, *Mean*, *Closest*, and GLVQ for the Mice (*left*) and USPS (*right*) data sets. GLVQ is evaluated using the maximum number of allocated prototypes and is included for reference. Plots are the average of the $K = 5$ validation results

Table 1 The number of epochs required to reach peak GLVQ baseline classification performance (value reported next to the data set name) with the resulting number of prototypes and cost reported for each data set and NSM method

Approach	Mice 84.24 %			USPS 94.47 %			LCVF 93.95 %		
	Epochs	Ptypes	Cost	Epochs	Ptypes	Cost	Epochs	Ptypes	Cost
PC	865	32	180.8	8414	66	1076	1036	45	110.2
Sampling cost	832	32	184.8	–	–	–	1349	48	103.0
Misclassified cost	731	31	190.8	9146	70	1107	1118	46	105.8
Mean	918	33	178.5	–	–	–	1108	46	107.3
Near-mean	704	31	195.4	–	–	–	1304	48	110.8
Boundary	822	32	196.6	–	–	–	1605	50.67	107.3

Table 2 Peak classification accuracy, number of prototypes, and total cost averaged over K-folds for each data set and NSM method

Approach	Mice			USPS			LCVF		
	Acc. (%)	Ptypes	Cost	Acc. (%)	Ptypes	Cost	Acc. (%)	Ptypes	Cost
PC	89.11	40	173.9	94.58	70	1065	95.57	57	96.77
Sampling cost	86.72	40	157.0	94.34	67	1080	94.97	66	89.94
Misclassified cost	88.06	40	168.1	94.47	70	1107	94.98	55	101.4
Mean	87.01	40	165.9	92.42	69	1337	94.97	61	99.29
Near-mean	88.06	39	168.1	94.09	70	1185	94.84	57	98.92
Boundary	85.40	40	186.5	90.70	69	1493	94.62	54	101.3
Full GLVQ	84.24	40	173.9	94.47	70	1057	93.95	70	94.88

3.3 Mice Protein Expression Results

As shown in Fig. 1 (left, top), all NSM methods surpass the classification accuracy of the baseline GLVQ and do so with fewer prototypes. Table 1 shows that the *Misclassified Cost* and *Near-Mean* are the first to reach this benchmark, utilizing the fewest prototypes (31 for each method). *Misclassified Cost* appears to dominate classification accuracy until approximately epoch 1200 and the insertion of the 36th prototype. From Fig. 1 (left, top) and seen in Table 2, beyond 1200 epochs *PC* offers the highest accuracy of 89.11 % with *Misclassified Cost* and *Near-Mean* finishing at 88.06 %. Table 2 further shows that *PC* achieves the highest cost, matching that of the baseline GLVQ.

The cost performance of the NSM methods is closely related to classification accuracy as is shown in Fig. 1 (left, bottom). This result is anticipated due to the formulation of GLVQ as a gradient descent algorithm. While all of the NSM methods show decreasing cost, *PC* shows the largest reduction followed by *Sampling Cost*. It is interesting to note that *Misclassified Cost* achieves higher classification accuracy even though several other NSM methods have lower cost curves. This difference is likely due to the fact that *Misclassified Cost* targets cost reduction by evaluating only misclassified samples for the placement of new prototypes. Addressing those misclassifications early on has the potential to strongly shape classification accuracy. Table 2 further shows that each method requires nearly the same number of prototype vectors to achieve their top accuracy, and that no clear trend exists between peak accuracy and associated cost.

3.4 USPS Handwritten Results and Discussion

Figure 1 (right, top) shows that *Misclassified Cost* has superior classification performance for the first 1500 epochs. Beyond 1500 epochs, the classification accuracy achieved by *PC* surpasses the other NSM method. In comparing the two direct cost minimization methods, the classification accuracy of *Sampling Cost* surpasses that of *Misclassified Cost* between 1500 and 7500 epoch. After 7500 epochs, the accuracy of *Misclassified Cost* is slightly better. Unlike the Mice Protein Expression data set, the baseline GLVQ performs on par or better than many of the NSM methods evaluated. This is further supported in Table 1 where we see that *PC* and *Misclassified Cost* are the only NSM methods that meet or exceed GLVQ for the USPS data set. The baseline GLVQ's strong performance might be attributed to the training time it enjoys with the full number of prototypes and that the number of prototypes used adequately represents the classification complexity of the data set. It is also possible that a non-dynamic GLVQ method is appropriate for relatively simple and "well balanced" classes (each class with approximately the same number of samples).

The two NSM methods that directly minimize cost offer some of the best classification performance. It interesting that *PC's* focus on the principal variance direction of the misclassified samples has the effect of indirectly minimizing the cost function yet leads to the lowest overall cost curve and highest classification accuracy as seen in Fig. 1 (right, bottom) and Table 2 respectively. The contextual information from clustering the misclassified samples with *PC* coupled with the "split metric" seems to provide superior initial prototype placement. Table 2 again shows that peak classification accuracy is achieved with approximately the same number of prototypes (*Sampling Cost* doing slightly better), and that the costs associated with those accuracies shows no clear trend.

3.5 LCVF Results and Discussion

Misclassified Cost offers very strong classification (Fig. 2 (top)) for the first half of the results (until approx epoch 1400), which is consistent with the Mice and USPS results. In Table 1, only *PC* achieves the baseline classification accuracy before *Misclassified Cost* with any noticeable lead. Up to 1200 epochs, *Misclassified Cost* also offers the strongest cost reduction as shown in Fig. 2 (bottom). After epoch 1400, the NSM methods (with exception of *PC* and *Boundary*) seem to converge, resulting in maximum classification accuracies in the range of 94.84–94.98 % (a difference of 0.14 %), which is also seen in Table 2. While the convergent result in the second half of Fig. 2 (top and bottom) may not be surprising due to the small sample size and disparate number of elements per class in the LCVF data set, *PC* does seem to offer marked performance gains, peaking at 95.57 % and resulting in the second lowest total cost. We see that *Sampling Cost* obtains a slightly lower cost than *PC*. Table 2 shows that widely varying numbers of prototypes are associated with the peak accuracies achieved by different NSM methods, while again there is no clear trend in resulting costs.

4 Summary

In this paper we promote the dynamic addition of prototype vectors to achieve superior performance and efficiency for GLVQ. We introduce the concept of network structure modification (NSM) into the standard LVQ taxonomy to describe individual methods for dynamic addition/deletion of prototypes within the network. This paper presents two new NSM methods, *Misclassified Cost* and *Near-Mean*, to achieve improved classification accuracy. The former explicitly minimizes the cost function by placing prototypes in way that minimizes the cost due to misclassified samples. *Near-Mean* selects the misclassified sample nearest the mean of the misclassified samples from the class with the largest classification error.

Several NSM methods are evaluated based on training time and prototypes required to meet a baseline classification performance of GLVQ. We find overwhelming evidence of improved classification accuracy with fewer prototype when considering a DGLVQ. We also find that over all three data sets, *Misclassified Cost* is consistently one of the better performing methods based on classification accuracy at the earliest opportunity. This fast convergence with fewer prototypes is a benefit in terms of overall performance and may support a reduced generalization error [6]. Faster convergence coupled with *Misclassified Cost's* diminishing computational complexity as training continues could be beneficial in real-time continuous learning applications.

We examine the trends of NSM methods as prototype networks expand and reach their peak performance configurations. Overall, we see strong classification accuracy from methods that effectively control cost, with some of the best performance from methods that minimize the cost directly. While *Sampling Cost* offers good classification and cost minimization performance by selecting candidate positions from randomly selected samples, *Misclassified Cost* offers promise as an alternative, with better classification accuracy shown for all data sets (including a full 2 % gain in the Mice Protein Expression data).

While we anticipated a clear distinct advantage of direct cost minimization NSM methods, *PC* indirectly minimizes cost and consistently results in the best accuracy and cost performance. Our adaptation and implementation of the *PC* method to dynamically add prototypes in GLVQ showed the best results across all data sets. *PC's* use of the misclassified sample variance allowed for a more informed prototype placement. This suggests future work related to cluster metrics to aid prototype placement is warranted. Using the same cluster metrics could improve prototype initialization, to include the number per class and their specific locations, which may result in improved accuracy and cost minimization performance, while reducing training requirements.

Acknowledgments The authors would like to thank Dr. E. Merényi for making the LCVF data set available for this work. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, the U.S. Department of Defense, or the U.S. Government.

References

1. Kohonen, T.: The self-organizing map. In: Proceedings of the IEEE, vol. 78, no. 9, pp. 1464–1480, Sept 1990
2. Losing, V., Hammer, B., Wersing, H.: Interactive Online Learning for Obstacle Classification on a Mobile Robot. IEEE (2015)
3. Kirstein, S., Wersing, H., Körner, E.: Rapid online learning of objects in a biologically motivated recognition architecture. In: German Pattern Recognition Conference DAGM, pp. 301–308 (2005)
4. Schleif, F.M., Villmann, T., Hammer, B.: Local metric adaptation for soft nearest prototype classification to classify proteomic data. In: International Workshop on Fuzzy Logic and Applications, Lecture Notes in Computer Science, vol. 3849, pp. 290–296. Springer (2006)
5. Mendenhall, M.J., Merényi, E.: Relevance-based feature extraction for hyperspectral images. IEEE Trans. Neural Netw. (2008)

6. Crammer, K., Gilad-bachrach, R., Navot, A., Tishby, N.: Margin analysis of the lvq algorithm. In: Advances in Neural Information Processing Systems, pp. 462–469. MIT press (2002)
7. Poirier, F.: DVQ: dynamic vector quantization application to speech processing. In: Second European Conference on Speech Communication and Technology, EUROSPEECH 1991, Genova, Italy, 24–26 Sept 1991
8. De Stefano, C., D'Elia, C., Marcelli, A., di Frecac, A.: Improving dynamic learning vector quantization. In: 18th International Conference on Pattern Recognition ICPR 2006, vol. 2, pp. 804–807 (2006)
9. Kohonen, T.: Self-organizing Maps, 3rd edn. Springer (2000)
10. Sato, A., Yamada, K.: Generalized learning vector quantization. In: Advances in Neural Information Processing Systems, pp. 423–429. The MIT Press (1996)
11. Grbovic, M., Vucetic, S.: Learning vector quantization with adaptive prototype addition and removal. In: Proceedings of the 2009 International Joint Conference on Neural Networks IJCNN'09, pp. 911–918. IEEE Press, Piscataway, NJ, USA (2009). http://dl.acm.org/citation.cfm?id=1704175.1704308
12. Bermejo, S., Cabestany, J., Payeras, M.: A new dynamic lvq-based classifier and its application to handwritten character recognition. In: ESANN, pp. 203–208 (1998)
13. Kietzmann, T.C., Lange, S., Riedmiller, M.: Incremental GRLVQ: learning relevant features for 3D object recognition. Neurocomput. **71**(13–15), 2868–2879 (2008). http://dx.doi.org/10.1016/j.neucom.2007.08.018
14. Zell, A., Mamier, G., Vogt, M., Mache, N., Hübner, R., Döring, S., Herrmann, K., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeorgiou, A., Posselt, D., Schreiner, T., Kett, B., Clemente G., Wieland J.: Stuttgart Neural Network Simulator (SNNS): User Manual, Version 4.1 (1995)
15. Lichman, M.: UCI Machine Learning Repository (2015). http://archive.ics.uci.edu/ml
16. Hull, J.J.: A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell. **16**(5), 550–554 (1994)
17. Merényi, E., Farrand, W., Taranik, J., Minor, T.: Classification of hyperspectral imagery with neural networks: comparison to conventional tools. EURASIP J. Adv. in Signal Process. **2014**(1), 71 (2014). http://asp.eurasipjournals.com/content/2014/1/71
18. Higuera, C., Gardiner, K.J., Cios, K.J.: Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. In: PLoS ONE **10**(6): e0129126 (2015)
19. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning. In: Springer Series in Statistics. Springer New York Inc., New York, NY (2009)
20. Merényi, E., Singer, R.B., Miller, J.S.: Mapping of spectral variations on the surface of mars from high spectral resolution telescopic images. Icarus **124**, 280–295 (1996)