Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 45, S. 8-13, Aralık 2022

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 45, pp. 8-13, December 2022

Research Article

Examining Variants of Learning Vector Quantizations According to Normalization and Initialization of Vector Positions

Rıfat Aşlıyan[1*]

[1*] Aydın Adnan Menderes Üniversitesi, Fen Fakültesi, Matematik Bölümü, Aydın, Türkiye, (ORCID: 0000-0003-1495-713X), rasliyan@adu.edu.tr

(1st International Conference on Engineering, Natural and Social Sciences ICENSOS 2022, December 20 - 23, 2022)

(**DOI:** 10.31590/ejosat.1222296)

ATIF/REFERENCE: Aşlıyan, R. (2022). Examining Variants of Learning Vector Quantizations According to Normalization and Initialization of Vector Positions. *European Journal of Science and Technology*, (45), 8-13.

Abstract

Learning Vector Quantization is a prototype-based artificial neural network. The classification is performed by representing the data set with the prototype vectors of the classes. In this study, using some variants of Learning Vector Quantization such as LVQ1, LVQ2.1, LVQ3, LVQX, and OLVQ1, the systems are designed and implemented, and they are examined according to initializations of prototype vectors and data sets. Every data set is divided into training and testing data sets. With the training data set, all LVQ networks are trained in a reinforcement learning strategy, and the models for each network are generated to test the success of the systems. In addition, the systems are compared with each other using some distinct normalization techniques such as z-score and linear scaling. In initial conditions, all prototype vectors can be randomly selected, and the values of all prototype vectors can be assigned to zero. The generated systems are evaluated by accuracy and f-measure benchmark measures and compared by their success rates.

Keywords: Learning Vector Quantization, LVQ1, LVQ2.1, LVQ3, LVQX, OLVQ1.

Normalizasyona ve Prototip Vektörlerin Başlangıç Değerlerine Göre Öğrenmeli Vektör Kuantalama Metotlarının İncelenmesi

Öz

Öğrenmeli Vektör Kuantalama, prototip tabanlı bir yapay sinir ağıdır. Öğrenmeli Vektör Kuantalama ile sınıflandırma, veri seti sınıfları, prototip vektörleri ile temsil edilerek gerçekleştirilir. Bu çalışmada, Öğrenmeli Vektör Kuantalama'nın LVQ1, LVQ2.1, LVQ3, LVQX ve OLVQ1 gibi bazı LVQ varyantları kullanılarak sistemler tasarlanmış, gerçekleştirilmiştir. Oluşturulan sistemler, veri setlerine ve prototip vektörlerinin başlangıç değerlerine göre incelenmiştir. Her veri seti eğitim ve test veri setlerine bölünmüştür. LVQ ağları destekleyici öğrenme stratejisi ile eğitim veri setini kullanarak eğitilir. Sistemlerin başarısını test etmek için her ağ için modeller oluşturulmuştur. Ayrıca sistemler, z-skoru ve doğrusal ölçekleme gibi bazı belirgin normalizasyon teknikleri kullanılarak birbirleriyle karşılaştırılır. Başlangıç değeri atamalarında, tüm prototip vektörleri için rastgele değerler seçilebilir ve tüm prototip vektörlerinin değerleri sıfıra atanabilir. Geliştirilen sistemler, doğruluk ve f-ölçüsü metrikleri ile değerlendirilmiştir ve başarı oranları ile karşılaştırılmıştır.

Anahtar Kelimeler: Öğrenmeli Vektör Kuantalama, LVQ1, LVQ2.1, LVQ3, LVQX, OLVQ1.

[*] Sorumlu Yazar: rasliyan@adu.edu.tr

1. Introduction

Learning Vector Quantization (LVQ) which has a competitive learning approach (Kohonen, 1986; Kohonen et al., 1988) is a prototype-based artificial neural network method proposed by Teuvo KOHONEN in 1986. It has a reinforcement learning and winner-takes-all strategy. The prototypes represented by the LVQ network map the training set. Every class has its prototypes, and the distances between a sample input of a dataset and prototype vectors are calculated using a selected distance metric, mostly Euclidean metric. The closest prototype vector to the input sample is the winner prototype vector, and the other vectors are the loser vectors. The output of the winner prototype vector is one, but the other prototype vectors' output is zero. The learning process of LVQ is to change the position of the winner prototype vector. Namely, the prototype vector is gotten closer to the sample input vector using a learning rate if the class of the prototype vector is the same as the class of the sample vector. Otherwise, the prototype vector is sent away from the sample input. This process is applied for all sample inputs until the maximum iteration has been reached or some specified conditions are satisfied. LVQ can be used for both binary and multi-class categorization. Each class can have its prototypes and the prototypes represent the categories.

With little distinct properties, there are some variants of LVQ in (McDermott, 1990; Makino et al., 1992; Kohonen, 1995). LVQ1 is the first developed LVQ network, and LVQ2 (Kohonen, 1990; Kohonen et al., 1996) network adjusts the borders of the classes and utilizes better positions of the prototype vectors. The variant of LVQ3 (Katagiri and Lee, 1993; Kohonen et al., 1996) network added a stabilizing constant to increase the accuracy for the later iterations. OLVQ1 (Kohonen, 1992) is a variant of LVQ which optimizes the network with an adjustment of the learning rate for each iteration. Optimized LVQ can be performed by other variants of the LVQ network. LVQX (Pham and Oztemel, 1993; Pham and Oztemel, 1994; Öztemel, 2012) variant brought new properties to LVQ as local and global winner prototypes.

There are some other LVQ variant techniques. LVQRKV (Günel et al., 2016), which proposed a geometrical scheme to prevent the problem of the generalized delta learning rule, adapts the prototype vectors by rotating the vectors on hyper-spheres in the training data set.

GLVQ (Sato and Yamada, 1995) updates the prototype vectors on the steepest descent technique to reduce the cost function.

In the variant of GRLVQ (Hammer and Villmann, 2002), a new scheme to enlarge GLVQ by weighting the variable of the sample input vectors was proposed.

In this work, for some data sets and distinct initial values of prototype vectors, some systems with LVQ variants are designed and implemented. According to these, the systems are examined and compared with each other.

This paper is organized as the following. In the next section, the variants of Learning Vector Quantizations such as LVQ1, LVQ2, LVQ2.1, LVQ3, LVQX, and OLVQ1 have been explained in short. The classification systems' results are given and discussed in the following section. The general points of this work are utilized in the conclusion section.

2. Material and Method

In this study, the variants of Learning Vector Quantization (LVQ) have been designed and implemented by some initialization of the prototype vectors. LVQ developed by Teuvo KOHONEN is one of the artificial neural network algorithms which learns from a training set in a reinforcement learning strategy.

LVQ is based on winner-takes-all and competitive learning. LVQ network is represented as prototype vectors (or codebook vectors). In the training stage, prototype vectors are placed in the most adequate positions in the training set. After that, with these prototype vectors, classification is applied according to the K-Nearest Neighbor approach. The strategy of LVQ learning is performed by moving prototype vectors closer or further to the sample vectors in the training set. LVQ prototype vectors try to find the best match to the training set. LVQ algorithm classifies the samples of not only binary classes but also multi-classes.

As shown in Figure 1, the topology of LVQ includes three layers as an input layer, one hidden layer (also called Kohonen layer), and an output layer. The input vector (a sample in the training set) $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is given to the input layer of LVQ. The neurons of the input layer are fully connected with the neurons of Kohonen layer. The directed edges are represented as weights from one neuron input layer to other neuron in Kohonen layer. For example, the weight w_{11} is a connection between the neuron x_1 and the neuron Y_1. The neuron Y_j can be thought of as a prototype vector. In other words, $Y_j = (w_{1j}, w_{2j}, \dots w_{nj})$. For determining the distance between the input vector and the prototype vectors, in general, Euclidean distance metric is used to find the winning prototype vector. In the Kohonen layer, the output of the winner neuron or winner prototype vector, y_j is 1, but the other neurons' outputs are zero. In the training process of LVQ, the only winner prototype vector gets closer to the input vector with the learning rate of the distance between them. These iterations go on for every input vector until the maximum iterations are reached or the minimum cost is obtained. Each class in the training set is served as these prototype vectors. After the distance between every sample in the test set and the prototype vectors are computed, as the Nearest Neighbor classifier, the input sample is categorized with the class of the prototype vector which is the nearest to it.

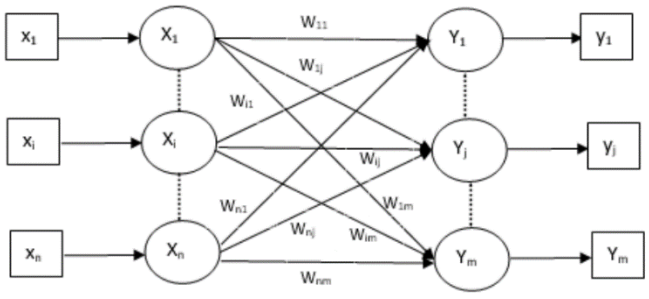

Figure 1. The structure of LVQ

The variant methods of Learning Vector Quantization are explained in short.

2.1.1. LVQ1

LVQ1 is the first variant network of Learning Vector Quantization. The following algorithm shows the training process of LVQ1. Basically, for every input vector, the winner prototype vector is detected, and if the class of the input vector equals the class of the winner prototype vector, the winner prototype vector is gotten closer to the input vector so that the input vector can win again in the next iterations.

LVQ1 Network Algorithm

1. Initialization of prototype vectors, namely weights, $Y_j = (w_{1j}, w_{2j}, \dots w_{nj})$.
2. Decide the number of prototype vectors for each class.
3. Determine the learning rate, $\delta \in (0,1)$, and the number of maximum epoch
4. For each x vector in the training set
5. For each prototype vector w
6. Calculate the distances between input vectors and prototype vectors with Euclidean distance as shown in Equation 1.

$$d(x, w) = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_i - w_{ij})^2 \quad (1)$$

where n is the number of input vectors in the training set and m is the number of prototype vectors.

End

7. Find the winner prototype vector w.
8. Update the winner prototype vector indicated in Equations 2 and 3.

$$w_{new} = w_{old} + \delta (x - w_{old}) \text{ if } P = C \quad (2)$$
$$w_{new} = w_{old} - \delta (x - w_{old}) \text{ if } P \neq C \quad (3)$$

where C is the class of the input vector, and P is the class of the winner prototype vector.

End

9. Decrease the learning rate, δ with a decreasing rate, $\beta \in (0,1)$ as shown in Equation 4.

$$\delta = \delta\beta \quad (4)$$

2.1.2. LVQ2 and LVQ2.1

LVQ2 network is constituted by a window is another variant of LVQ. LVQ2 is developed to improve the success of the standard LVQ model. In particular, LVQ2 network tries to prevent the classification error in the boundary values of the classes. Apart from standard LVQ, during the training, this network proposes to change the two closest prototype vectors to the input vector at the same time.

If these two prototype vectors are called $w1$ and $w2$, their weights must be updated when the following conditions are satisfied.

Condition 1: The prototype vector, $w1$ is the closest weight vector to the input vector, and the other prototype vector, $w2$ is the next closest vector to it. The class of input vector is different from the class of $w1$ and is the same class of $w2$.

Condition 2: The input vector is in the specified range, namely in a window. The window size is between 0 and 1, but it is chosen between 0.2 and 0.3 in general.

For LVQ2.1, condition 1 is a little bit different. Case 1: The class of input vector is different from the class of $w1$ and is the same class of $w2$. Case 2: The class of input vector is different from the class of $w2$ and is the same class of $w1$. Condition 1 is satisfied when one of the two cases above is true. In LVQ2.1, the update operations are performed in this way. The prototype vector with the same class of the input vector is gotten closer to the input vector, The prototype vector which is distinct from the class of the input vector is gotten further from the input vector.

If the two conditions are satisfied, the prototype vectors $w1$ and $w2$ weights are updated as displayed in Equations 5 and 6. $w1_{new}$ and $w2_{new}$ vectors represent new values of the prototype vectors. In the same way, $w1_{old}$ and $w2_{old}$ vectors stand for old values of the prototype vectors.

$$w1_{new} = w1_{old} - \delta (x - w1_{old}) \quad (5)$$

$$w2_{new} = w2_{old} + \delta (x - w2_{old}) \quad (6)$$

Whether or not the input vector is in the specified window, is determined by Equations 7 and 8.

$$c = (1 - \text{window})/(1 + \text{window}) \quad (7)$$

$$\text{if } \min(d_1/d_2, d_2/d_1) > c \quad (8)$$

Where d_1 is the distance between the input vector and the prototype vector $w1$, and d_2 is the distance between the input vector and the prototype vector $w2$.

The condition in Equation 8 is satisfied, and the input vector is in the window.

2.1.3. LVQ3

LVQ3 network applies all LVQ2 operations. In Addition, LVQ3 includes some extra update operations when the prototype vectors $w1$, $w2$, and the input vector are in the same class. As we described in LVQ2 network, the prototype vectors $w1$ and $w2$ are the closest two vectors to the input vector in the training set. As shown in Equations 9 and 10, the prototype vectors $w1$ and $w2$ are gotten closer to the input vector when all three vectors are in the same category, and the input vector is in the specified window of the two prototype vectors.

$$w1_{new} = w1_{old} + \varepsilon\delta (x - w1_{old}) \quad (9)$$

$$w2_{new} = w2_{old} + \varepsilon\delta (x - w2_{old}) \quad (10)$$

Where ε is a stabilizing constant. $\varepsilon \in (0,1)$. It is preferred to become between 0.1 and 0.5.

2.1.4. LVQX

In LVQ2 network, the weights of both two prototype vectors are rarely updated, however, in LVQX network, both prototype vectors' weights are changed in every iteration. This increases the learning speed of the network and the generalization ability and decreases the learning time. The LVQX network designates two prototype vectors, the global winner and the local winner. The global winner prototype vector is the prototype vector that has the minimum distance to the input vector in the training set. But, the local winner prototype vector is the prototype vector that has the minimum distance between the input vector and the prototype vectors of the input vector's class.

According to the learning strategy of LVQX network, if the global winner and the local winner prototype vectors are the same, then only this prototype vector is updated as displayed in Equation 5. But, if the global winner and the local winner prototype vectors are different, the global winner vector is moved away from the input vector, and the local winner vector gets closer to the input vector as shown in Equations 6 and 7.

If w_g equals w_l then

$$w_{g,new} = w_{g,old} + \delta\,(x - w_{g,old}) \quad (11)$$

else

$$w_{g,new} = w_{g,old} - \delta\,(x - w_{g,old}) \quad (12)$$

$$w_{l,new} = w_{l,old} + \delta\,(x - w_{l,old}) \quad (13)$$

Where w_g is the global winner prototype vector, and w_l is the local winner prototype vector.

2.1.5. OLVQ1

OLVQ1 network is a variant of LVQ in which the learning rate is adapted for each iteration. The learning rate, δ is updated as displayed in Equation 14.

$$\delta(t) = \frac{\delta(t-1)}{1+s(t)\delta(t-1)} \quad (14)$$

Where t is the iteration value and $s(t) = +1$ if the winning prototype vector has the same class as the input vector, otherwise $s(t) = -1$.

3. Result and Discussion

In this study, the systems based on the variants of LVQ networks as LVQ1, LVQ2, LVQ2.1, LVQ3, OLVQ1, and LVQX are designed and implemented. The successes of the systems are evaluated by benchmark metrics such as accuracy and f-measure. These metric values are calculated as shown in Equations 14, 15, 16, and 17.

Accuracy and f-measure have been used to evaluate the success of the classification systems. The f-measure is calculated by combining the Recall and Precision evaluation measures in one equation, as seen in Equations 15, 16, 17, and 18 below. The f-measure for different classes has been generalized by the Macro average of the f-measures. The accuracy values of the systems are measured as shown in Equation 17.

TP: True Positive means the correctly predicted positive values.

TN: True Negative means the correctly predicted negative values.

FP: False Positive means the incorrectly predicted negative values.

FN: False Negative means the incorrectly predicted positive values.

$$\text{Precision} = \text{TP}/\text{TP} + \text{FP} \quad (15)$$

$$\text{Recall} = \text{TP}/\text{TP} + \text{FN} \quad (16)$$

$$\text{Accuracy} = \text{TP} + \text{TN}/\text{TP} + \text{FP} + \text{FN} + \text{TN} \quad (17)$$

$$\text{F} - \text{measure} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision}) \quad (18)$$

The systems of the variants of LVQ have been implemented with Matlab and the computer with the features as Intel Core i7 2.40 GHz CPU, 16 GB RAM, and 64-bit Windows 10 Operating System.

In the developed systems, iris (Iris data set, 2022) and wine (Wine data set, 2022) have been used for testing the systems.

The iris data set includes three categories of fifty samples for every class. Classes are represented as a type of iris plant. This data set has 150 samples, and the values of attributes are real numbers. The wine data set also contains three categories. But, the attributes have thirteen real or integer values, and the total sample size is 178.

In this study, the five different variants of LVQ LVQ1, LVQ2.1, LVQ3, OLVQ1, and LVQX have been implemented with learning rate=0.1, stabilizing constant=0.001, 10 class prototypes, and maximum iterations=30, and as shown in Tables 1-10, the success rates of them are displayed according to two data sets, initialization conditions of prototype vectors and some normalizations of feature vectors.

Before training the LVQ networks, their prototype vectors must be initialized by some numbers. Two conditions have been analyzed in this work zero initialization of them and random initialization between zero and one. It is also examined for normalization of the features in the data sets. In addition, the results are compared according to the normalization conditions. All results are obtained by the z-score, linear scale, and without normalization.

In the following tables, P represents the prototype vectors, and P=0-1: Initialization P between 0 and 1 randomly. P=0: Initialization of all P to zero.

Evaluating the system, the data set is divided into a train set and a test set. Five different train sets and test sets are generated by randomly selecting samples from the data set. Hence, an LVQ model is constituted with each train set. As displayed in the following tables, the accuracy and f-measure values are the means of accuracy and macro f-measure from the train sets. Their standard deviations are presented in the tables.

According to the success of the variant methods of LVQ, the most successful method is LVQX for both iris and wine data sets. The accuracy rates are approximately 97% accuracy and f-measure rates in the iris data set. However, in the wine data set, 98% accuracy and 97% f-measure values have been computed. The success rates of the other variants are similar to each other.

If the systems are evaluated by normalization, it can be said that normalization has highly increased the systems' success, and the best normalization is linear scaling in both data sets.

As shown in the tables, when normalizing with a linear scale, the models have not been affected much by the initializations. However, when the models without normalization are more successful when they initialize their prototype vectors between zero and one randomly.

Table 1. The success rates of the iris data set for LVQ1

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.83±0.154	0.81±0.154
	Z-score	0.89±0.21	0.86±0.024
	Linear scaling	**0.96±0.026**	**0.95±0.029**
P=0	No Nor.	0.77±0.207	0.72±0.211
	Z-score	0.88±0.004	0.88±0.046
	Linear scaling	0.95±0.019	0.94±0.022

Table 2. The success rates of the iris data set for LVQ2.1

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.85±0.142	0.85±0.145
	Z-score	0.92±0.022	0.914±0.030
	Linear scaling	0.94±0.026	0.93±0.027
P=0	No Norm.	0.72±0.184	0.65±0.196
	Z-score	0.89±0.026	0.88±0.025
	Linear scaling	**0.95±0.017**	**0.94±0.017**

Table 3. The success rates of the iris data set for LVQ3

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.95±0.009	0.95±0.011
	Z-score	0.90±0.013	0.90±0.013
	Linear scaling	**0.96±0.024**	**0.96±0.022**
P=0	No Norm.	0.84±0.136	0.83±0.136
	Z-score	0.87±0.048	0.86±0.053
	Linear scaling	0.95±0.020	0.94±0.021

Table 4. The success rates of the iris data set for OLVQ1

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.74±0.071	0.72±0.073
	Z-score	0.92±0.023	0.91±0.020
	Linear scaling	**0.96±0.018**	**0.95±0.018**
P=0	No Norm.	0.67±0.013	0.67±0.016
	Z-score	0.87±0.014	0.87±0.013
	Linear scaling	0.93±0.021	0.92±0.020

Table 5. The success rates of the iris data set for LVQX

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.95±0.008	0.94±0.030
	Z-score	0.90±0.032	0.90±0.031
	Linear scaling	**0.97±0.008**	**0.97±0.012**
P=0	No Nor.	0.96±0.008	0.95±0.010
	Z-score	0.88±0.020	0.87±0.017
	Linear scaling	0.97±0.038	0.96±0.008

Table 6. The success rates of the wine data set for LVQ1

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.49±0.044	0.48±0.044
	Z-score	0.96±0.022	0.95±0.023
	Linear scaling	0.97±0.95	0.96±0.021
P=0	No Norm.	0.48±0.071	0.44±0.062
	Z-score	0.96±0.010	0.95±0.009
	Linear scaling	**0.98±0.016**	**0.97±0.016**

Table 7. The success rates of the wine data set for LVQ2.1

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.60±0.062	0.59±0.064
	Z-score	0.95±0.037	0.94±0.038
	Linear scaling	**0.98±0.010**	**0.98±0.012**
P=0	No Norm.	0.54±0.068	0.52±0.080
	Z-score	0.96±0.011	0.95±0.013
	Linear scaling	0.98±0.017	0.97±0.019

Table 8. The success rates of the wine data set for LVQ3

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.57±0.058	0.54±0.067
	Z-score	0.93±0.038	0.92±0.044
	Linear scaling	**0.98±0.007**	**0.98±0.010**
P=0	No Norm.	0.56±0.055	0.53±0.054
	Z-score	0.96±0.011	0.95±0.013
	Linear scaling	0.98±0.010	0.97±0.013

Table 9. The success rates of the wine data set for OLVQ1

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.56±0.023	0.54±0.014
	Z-score	0.91±0.034	0.90±0.036
	Linear scaling	**0.98±0.004**	**0.97±0.004**
P=0	No Norm.	0.50±0.063	0.47±0.056
	Z-score	0.95±0.004	0.95±0.010
	Linear scaling	0.97±0.003	0.97±0.007

Table 10. The success rates of the wine data set for LVQX

Init.	Normalization	Accuracy	F-measure
P=0-1	No Norm.	0.99±0.009	0.69±0.035
	Z-score	0.97±0.024	0.96±0.24
	Linear scaling	**0.99±0.009**	**0.99±0.12**
P=0	No Norm.	0.69±0.029	0.68±0.40
	Z-score	0.96±0.007	0.96±0.011
	Linear scaling	0.99±0.012	0.98±0.011

4. Conclusions and Recommendations

In this study, using some variants of Learning Vector Quantization such as LVQ1, LVQ2.1, LVQ3, LVQX, and OLVQ1, the systems are designed and implemented, and they are examined according to initializations of prototype vectors and data sets. Every dataset is divided into training and testing data sets. With the training dataset, all LVQ networks are trained in a reinforcement learning manner, and the models for each network are generated to test the success of the systems. In addition, the systems are compared with each other using some distinct normalization techniques such as z-score and linear scaling. In initial conditions, all prototype vectors are randomly selected between zero and one, the values of all prototype vectors are assigned to zero. The generated systems are evaluated by accuracy and f-measure benchmark measures and compared by their success rates. This study shows LVQX network outperforms the other LVQ variants, and it can be said that linear scaling normalization improves the networks' success.

For the next studies, it is planned to use other variants of the LVQ method and the hybrid of these LVQ variants.

References

Günel, K., Aşlıyan, R. and İclal, G. (2016). A Geometrical Modification of Learning Vector Quantization Method for Solving Classification Problems. *Suleyman Demirel University Journal of Natural and Applied Sciences*, vol. 20(3), pp. 414-420.

Hammer, B. and Villmann, T. (2002). Generalized relevance learning vector quantization. *Neural Networks*, vol. 15(8-9), pp. 1059-1068.

Iris data set. (2022). Website [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris

Katagiri, S. and Lee, C.H. (1993). A new hybrid algorithm for speech recognition based on HMM segmentation and learning vector quantization. *IEEE Transactions on Speech and Audio Processing*, vol. 1(4), pp. 421-430.

Kohonen, T. (1986). Learning vector quantization for pattern recognition. Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland.

Kohonen, T., Barna, G. and Chrisley, R. (1988). Statistical pattern recognition with neural networks: Benchmarking studies. In *Proc. of the International Conference on Neural Networks (ICNN)*, vol. I, Los Alamitos, CA. IEEE Computer Soc. Press, p. 61-68.

Kohonen, T. (1990). Improved versions of learning vector quantization. In *Pro. of the International Joint Conference on Neural Networks (IJCNN)*, vol. 1, pages 545-550, San Diego, California.

Kohonen, T. (1992). New developments of learning vector quantization and self-organizing map. In *Proc. Symposium on Neural Networks*, Alliances and Perspectives in Senri, Osaka, Japan.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer, Berlin, Germany.

Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J. and Torkkola, K. (1996). LVQ_PAK: the learning vector quantization programming package. Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland.

Makino, S., Endo, M., Sone, T. and Kido, K. (1992). Recognition of *phonemes* in continuous speech using a modified LVQ2 method. *J. Acoustical Society of Japan*, vol. 13(6) pp. 351-360.

McDermott, E. (1990). LVQ3 for phoneme recognition. In *Proc. Spring Meet. Acoust. Soc. Jpn.*, p. 151-152.

Öztemel, E. (2012). *Yapay Sinir Ağları*, Ezgi Kitapevi, Bursa.

Pham, D.T. and Oztemel, E. (1993). Control Chart Pattern Recognition Using Combinations of Multilayer Perceptrons and Learning Vector Quantization Neural Networks. *Proc. Instn. Mech. Engrs*. Vol. 207, pp. 113-118.

Pham, D.T. and Oztemel, E. (1994). Control Chart Pattern Recognition Using Combinations of Multilayer Perceptrons and Learning Vector Quantization Neural Networks. *International Journal of Production Research*, vol. 32, 721-729.

Sato, A. and Yamada, K. (1995). Generalized Learning Vector Quantization", *NIPS*.

Wine data set. (2022). Website [Online]. Available: https://archive.ics.uci.edu/ml/datasets/wine