Learning Vector Quantization for Ordinal Classification: p-OGMLVQ

- Research Internship -
Project Report

Zhuoyun Kan (s3346935)

University of Groningen
Faculty of Science and Engineering

-

1 Introduction

Many classification algorithms focus on classifying data into classes with order information. These tasks are known as ordinal classification or ordinal regression, which have attracted much attention in recent years [1, 2, 3]. Different from the binary or nominal classification, the class order in ordinal classification tasks reflects certain relationships between categories, which is commonly seen in many real-world applications, e.g., corporate credit scoring [4], epidemiologic data [5] and student satisfaction [6].

The concept of learning vector quantization (LVQ) was introduced by Kohonen [7, 8]. It is a supervised learning algorithm which focuses on prototype-based classification problems. In the LVQ approach, classifiers are represented by a set of labelled prototypes, which indicate different classes in the same space of input examples. After the training phase, the class of an unlabelled example is determined by its nearest prototype based on a distance measure. The idea of assigning an input example to the class of the closest prototype is closely related to k-nearest neighborhood classification (k-NN) [9], but LVQ has advantages regarding storage memory and computational effort.

Many variants of the basic LVQ method have been proposed, aiming at improving the algorithm to achieve better classification results, e.g., generalized LVQ (GLVQ) [10] and generalized relevance LVQ (GRLVQ) [11]. However, in these approaches, the Euclidean distance between prototypes and training examples could result in problems, especially in high-dimensional datasets, since all input dimensions are weighted equally in this case. Therefore, metric adaption techniques were introduced to learn discriminative distance measures from training examples in matrix LVQ (MLVQ) [12], which contain updates for both prototypes and metric. MLVQ was further extended to generalized matrix LVQ (GMLVQ) [12, 13], which optimized the hypothesis margin and achieved a better generalization ability.

As an extension of GMLVQ, pair ordinal generalized matrix learning vector quantization (p-OGMLVQ) was proposed in 2012 [1]. It focuses on ordinal classification problems and has been further extended to accumulative ordinal generalized matrix learning vector quantization (a-OGMLVQ) in 2017 [3], which modifies the cost function of p-OGMLVQ and takes global class order into consideration.

This study aims to implement the p-OGMLVQ algorithm as a realization of ordinal classification in GMLVQ and apply it to real-world datasets to improve classification results concerning order information. Moreover, the performance of the algorithm is investigated in a number of example problems, including imbalanced datasets, limited feature vectors, global optimalities and so forth.

2 Background

Learning vector quantization (LVQ) is a prototype-based classification algorithm which was first proposed by Kohonen [7]. This approach enjoys high popularity as it is intuitive, easy to implement and is a natural tool for multi-class classification problems. In this section, the basic LVQ scheme and its extensions will be briefly introduced.

2.1 LVQ1

The original version of the learning vector quantization algorithm is known as LVQ1 [7]. Consider a training dataset $X = \{(x_i, y_i)|\mathbb{R}^m \times \{1, ..., C\}|\}_{i=1}^n$, where m is the dimension of the inputs, n is the number of training examples and C is the number of different classes. A typical LVQ classifier consists of L prototypes $\omega_j \in \mathbb{R}^m, j = 1, 2, ..., L$, where prototypes ω_j are labelled by $c(\omega_j) \in \{1, ..., C\}$. The classification procedure follows a winner-takes-all scheme. For an input data point $x \in \mathbb{R}^m$, the output class of x is determined by its closest prototype ω_i, $i = argmin_j d(x, \omega_j)$, where $d(x, \omega_j)$ is a distance measure in $\mathbb{R}^m$. Each prototype ω_j with class label $c(\omega_j)$ represents a receptive field in the input space so that a set of data points in this field will be assigned $c(\omega_j)$ by the LVQ classifier.

In the training phase of the LVQ algorithm, for each training example x_i with label $c(x_i)$, the closest prototype is adapted depending on the classification result. The closest prototype with the same class label as x_i will be attracted by x_i, while the closest prototype with a different class label will be pushed away from the example x_i. The objective of the learning procedure is to adapt prototypes automatically in the feature space so that the highest classification accuracy on novel data can be achieved.

2.2 Matrix LVQ

As an extension of the basic LVQ scheme, matrix LVQ (MLVQ) [12] introduced matrix learning in LVQ based on the given classification task. In the MLVQ algorithm, a full matrix distance measure is used, instead of the squared Euclidean distance $d(x, \omega) = (x - \omega)^T \cdot (x - \omega)$, because the use of the Euclidean distance can be problematic in case of high-dimensional or heterogeneous datasets; it is based on the implicit assumption that data can be represented by isotropic clusters. Thus, a generalized form of the Euclidean distance is defined as

$$d^{\Lambda}(x, \omega) = (x - \omega)^T \cdot \Lambda \cdot (x - \omega), \tag{1}$$

where Λ is a full $m \times m$ symmetric and positive-definite matrix, which indicates the correlations between features. The positive definiteness of Λ can be achieved by substituting

$$\Lambda = \Omega^T \Omega, \tag{2}$$

where $\Omega \in \mathbb{R}^{m \times m}$ is an arbitrary full-rank matrix. Also, Λ needs to be normalized after each learning epoch to prevent the learning algorithm from degeneration by enforcing

$$\sum_i \Lambda_{ii} = 1. \tag{3}$$

Therefore, the distance metric d^{Λ} corresponding to the squared Euclidean distance in a new coordinate system can be rewritten as

$$d^{\Lambda}(x, \omega) = [(x - \omega)^T \Omega^T][\Omega(x - \omega)] = [\Omega(x - \omega)]^2. \tag{4}$$

2.3 Generalized Matrix LVQ

Generalized LVQ (GLVQ) was first introduced in 1996 [10] using the steepest descent method based on an explicit cost function. In the training phase, for each data point x_i, a prototype pair is updated so that a convergence condition can be reached. The prototype pair consists of the closest correct and incorrect prototype. In the learning step, the closest correct prototype is dragged towards x_i, while the closest wrong prototype is pushed far away from x_i.

Extended from GLVQ, generalized matrix LVQ (GMLVQ) uses the adaptive full matrix distance measure in Eq. (1). Hence, the modified cost function is given as

$$f_{GMLVQ} = \sum_{i=1}^{n} \phi(\mu_i), \quad with \quad \mu_i = \frac{d^{\Lambda}(x_i, \omega^+) - d^{\Lambda}(x_i, \omega^-)}{d^{\Lambda}(x_i, \omega^+) + d^{\Lambda}(x_i, \omega^-)}, \tag{5}$$

where ϕ is a monotonically increasing function, e.g., the identity function $\phi(x) = x$ or the logistic function $\phi(x) = 1/(1 + e^{-x})$; $d^{\Lambda}(x_i, \omega^+)$ and $d^{\Lambda}(x_i, \omega^-)$ constitute the distances of the input pattern x_i to the closest correct and incorrect prototype. The term μ_i is the relative difference distance. The numerator is smaller than 0 if the classification of the input pattern is correct. The smaller the numerator, the larger the hypothesis margin of the classifier. The larger the margin, the more robust the classification results. The denominator scales the argument of ϕ so that it falls in the range [-1, 1]. The learning rules are given by taking the derivatives of the cost function with respect to the prototypes ω and the distance matrix Ω.

3 Pair Ordinal GMLVQ

In this section, the methodologies and the algorithm of pair ordinal GMLVQ (p-OGMLVQ) proposed by S. Fouad and P. Tiňo [1] for ordinal classification are presented in detail.

3.1 Ordinal GMLVQ classifiers

In the original GMLVQ methods [12, 13], only one single prototype pair (the closest correct and wrong prototypes) is updated in each training phase. However, unlike in the nominal classification, p-OGMLVQ modifies the original GMLVQ by extending the update rules of correct and incorrect prototypes with order information.

As an extended version of GMLVQ, p-OGMLVQ mainly has two differences from GMLVQ. First, Hebbian updates are applied to all qualified prototype pairs in p-OGMLVQ instead of the closest prototype pair. Besides, the cost function is a weighted sum version of the original GMLVQ cost function as given in Eq. (5) . It is composed of the weighted distances of both correct and incorrect prototypes from input data points. The weights for correct and incorrect prototypes are defined by a Gaussian weighting scheme, based on the ordinal class information.

3.2 Prototype classes

In p-OGMLVQ, the correct and incorrect prototype classes are identified by the label distance from the true class using an absolute error loss function and a ranking loss threshold. In particular, the p-OGMLVQ approach accepts a tolerable error on prototype classes. In the following, the error loss function and the ranking loss threshold are described in detail.

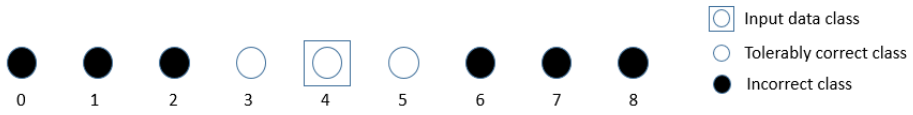


Figure 1: Identification of correct and incorrect prototype classes with rank loss threshold $L_{min} = 1$. The class of the input data point x_i is 4, indicated with a square. Prototype classes within the range of the threshold are viewed as correct ones (i.e., 3,4 and 5), while other prototype classes are regarded as incorrect ones (i.e., 0-2 and 6-8) indicated with black circles.

- **Absolute error loss function**

  Given a training pattern x_i with class label $c(x_i)$ and all the prototypes ω_j, the correct and incorrect prototype classes are classified by an absolute error loss function $H(c(x_i), c(\omega_j))$ [14] according to the ordinal labels

$$H(c(x_i), c(\omega_j)) = |c(x_i) - c(\omega_j)|. \tag{6}$$

- **Ranking loss threshold**

  A ranking loss threshold L_{min} [1] is set as a boundary to define correct and incorrect classes. If the loss function $H(c(x_i), c(\omega_j))$ is smaller than or equal to the threshold L_{min}, the class prototypes will be classified as tolerably correct; otherwise, the prototypes will be viewed as incorrect.

  As illustrated in Figure 1, the order information is used in the selection of prototypes from different categories. Therefore, if prototypes are spatially close to the training instance x_i, but the classes of the prototypes $c(\omega_j))$ are far away from the target class $c(x_i)$ (in terms of class order), the prototypes ω_j are considered as incorrect ones. The sets of correct and wrong classes can be written as

$$N(c(x_i))^+ = |c(x_i) - c(\omega_j)| \leq L_{min}, \tag{7}$$

$$N(c(x_i))^- = |c(x_i) - c(\omega_j)| > L_{min}, \tag{8}$$

  where $c(\omega_j)$ ranges from 1 to C.

3.3 Class prototypes to be adapted

Given a training instance x_i, the class prototypes to be adapted are selected in a different way from original GMLVQ method, and constitute two prototype groups ($W(x_i)^+$ and $W(x_i)^-$) based on the labels in $N(x_i)^+$ and $N(x_i)^-$.

- **Set of correct prototypes**

  For classes in $N(x_i)^+$, only the nearest prototype from each correct class will be selected and pushed toward x_i. The set of correct prototypes is defined as

$$W(x_i)^+ = \{\omega_{z(k)} | c(\omega_{z(k)}) = k \in N^+, z(k) = arg \min_{l \in W(k)} [d^\Lambda(x_i, \omega_l)]\}, \tag{9}$$

  where d^Λ is the distance metric introduced in Eq. (1).

- **Set of incorrect prototypes**

  Different from the set of correct prototypes, all incorrect prototypes in the neighborhood of x_i with labels in $N(x_i)^-$ are tended to be pushed away. The set of incorrect prototypes is given by

$$W(x_i)^- = \{\omega_z | c(\omega_z) = k \in N^-, d^\Lambda(x_i, \omega_z) < D\}, \tag{10}$$

  where D is the radius of the sphere neighborhood. In the following experiments, D is set as the median metric distance of input example x_i to all incorrect prototypes.

In p-OGMLVQ, prototypes to be updated are constructed in pairs. To form these pairs, prototypes from both sets ($W(x_i)^+$ and $W(x_i)^-$) are sorted in increasing order in terms of distance to the input example x_i. The number of prototype pairs r is given by $min(|W(x_i)^+|, |W(x_i)^-|)$, where $|W|$ represents the number of prototypes. The first $r \geq 1$ prototypes in both correct and incorrect set constitute r prototype pairs, which will be updated in the training phase.

3.4 Weighting scheme

Different from nominal LVQ classification, multiple pairwise prototypes are adapted in p-OGMLVQ. Although these correct and incorrect prototypes will be dragged towards and pushed away from input x_i, it makes sense that they are not updated to the same extent. In [1], these prototypes are trained following a weighting scheme with respect to the class distance from input x_i. The attraction and repulsion between prototypes and x_i will decrease and increase, respectively, with growing class difference $H(c(x_i), c(\omega_j))$. In addition, the weights for incorrect prototypes will diminish with increasing metric distance from the input example x_i.

- **Weights for correct prototypes**

  Given a training instance x_i, weights for correct prototypes $\omega_j \in W(x_i)^+$ are calculated by a Gaussian kernel

$$\alpha_j^+ = exp\Big\{ - \frac{H(c(x_i), c(\omega_j^+))^2}{2\sigma^2} \Big\}, \tag{11}$$

  where σ is the bandwidth of the Gaussian weighting scheme. A correct prototype in the class far away from the class of instance x_i in terms of label space has a small weight α_j^+, indicating a small update step size.

- **Weights for incorrect prototypes**

  For incorrect prototypes $\omega_j \in W(x_i)^-$, the weight factor α_j^- is determined as

$$\alpha_j^- = exp\Big\{ - \frac{(E_{max} - H(c(x_i), c(\omega_j^-)))^2}{2\sigma^2} \Big\} \cdot exp\Big\{ - \frac{d^\Lambda(x_i, \omega_j^-)}{2\sigma'^2} \Big\}, \tag{12}$$

  where E_{max} denotes the maximum absolute rank loss error between x_i and prototypes within the set $W(x_i)^-$, and $d^\Lambda(x_i, \omega_j^-)$ is the square metric distance between x_i and ω_j^-.

  σ' is the Gaussian kernel width for the distance $d^\Lambda(x_i, \omega_j^-)$.

3.5 Cost function

The cost function of p-OGMLVQ is extended from original GMLVQ. It scales the metric distance by multiplying weighting factors (α^+ and α^-) according to prototype pairs. For each prototype pair j, the cost function is formally defined as

$$\mu_j(x_i) = \frac{\alpha_j^+ \cdot d^\Lambda(x_i, \omega_j^+) - \alpha_j^- \cdot d^\Lambda(x_i, \omega_j^-)}{\alpha_j^+ \cdot d^\Lambda(x_i, \omega_j^+) + \alpha_j^- \cdot d^\Lambda(x_i, \omega_j^-)}. \tag{13}$$

The overall p-OGMLVQ cost function for one complete training phase is given as follows:

$$f_{OGMLVQ} = \sum_{i=1}^{n} \sum_{j=1}^{r} \phi(\mu_j(x_i)), \tag{14}$$

where r is the number of prototype pairs, and n is the number of training examples. ϕ is a monotonically increasing function, e.g., the identity function $\phi(x) = x$ or the logistic function $\phi(x) = 1/(1 + e^{-x})$. The identity function is applied in this study. Ideally, the value of the overall cost function will decrease with increasing training epochs and stop at a global minimum value. However, the descent procedures frequently end up in local minima as the cost function can have plenty of local minima that are far from the global optimal.

3.6 Update rules of prototypes and the metric

Steepest descent method is used for updating prototypes and the metric parameter in this study. The update rules for the p-OGMLVQ algorithm are derived from the derivatives of the cost function with respect to prototypes ω_j^+ and ω_j^+, and the metric Ω, which are given by

$$\Delta_{\omega_j^+} = 2 \cdot \eta_\omega \cdot \gamma_j^+ \cdot \Lambda \cdot (x_i - \omega_j^+), \tag{15}$$

$$\Delta_{\omega_j^-} = -2 \cdot \eta_\omega \cdot \gamma_j^- \cdot \Lambda \cdot (x_i - \omega_j^-), \tag{16}$$

$$\Delta_\Omega = -2 \cdot \eta_\Omega \cdot \{\gamma_j^+ \cdot \Omega \cdot (x_i - \omega_j^+)(x_i - \omega_j^+)^T - \gamma_j^- \cdot \Omega \cdot (x_i - \omega_j^-)(x_i - \omega_j^-)^T\}, \tag{17}$$

where γ_j^+ and γ_j^- are given as

$$\gamma_j^+ = \frac{2 \cdot \alpha_j^+ \cdot \alpha_j^- \cdot d^\Lambda(x_i, \omega_j^-)}{(\alpha_j^+ \cdot d^\Lambda(x_i, \omega_j^+) + \alpha_j^- \cdot d^\Lambda(x_i, \omega_j^-))^2}, \tag{18}$$

$$\gamma_j^- = \frac{1 - d^\Lambda(x_i, \omega_j^-)}{2 \cdot \sigma'^2} \cdot \frac{2 \cdot \alpha_j^+ \cdot \alpha_j^- \cdot d^\Lambda(x_i, \omega_j^+)}{(\alpha_j^+ \cdot d^\Lambda(x_i, \omega_j^+) + \alpha_j^- \cdot d^\Lambda(x_i, \omega_j^-))^2}, \tag{19}$$

η_ω and η_Ω are the learning rates for prototypes and metric, respectively.

3.7 Learning rate

Learning rates $0 < \eta_\omega < 1$ and $0 < \eta_\Omega < 1$ are important parameters in the algorithm, which control the size of learning steps in the training phase. Both η_ω and η_Ω are monotonically decreased with growing training epochs [12], and stated as

$$\eta_\omega(t) \leftarrow \frac{\eta_\omega(t-1)}{1+\tau(t-1)}, \tag{20}$$

$$\eta_\Omega(t) \leftarrow \frac{\eta_\Omega(t-1)}{1+\tau(t-1)}, \tag{21}$$

where $\tau > 0$ is the decay factor determining the annealing speed and t indicates the current training epoch.

3.8 P-OGMLVQ Algorithm

The overall p-OGMLVQ algorithm is summarized as follows:
1. **Initialization**:

   (1) Initialize prototype positions $\omega_q \in R^m$.

   (2) Initialize metric parameter Ω by setting it as the identity matrix.

2. **While** a stopping criterion in not reached, do:

   (1) Randomly select a training example x_i with class c_i

   (2) Determine the correct classes N^+ and wrong classes N^- for x_i, according to Eq. (7) and (8).

   (3) Determine the set of correct prototypes W^+ and the set of incorrect prototypes W^- based on Eq. (9) and (10), respectively.

   (4) Sort prototypes in W^+ and W^- in increasing order with respect to the distance to x_i.

   (5) Compute the weights α^+ and α^- for selected correct and incorrect prototypes, using Eq. (11) and (12), respectively.

   (6) Set $W^\pm = W(x_i)^\pm$, $r = 0$

   While ($W^+ \neq \emptyset$ and $W^- \neq \emptyset$), do:

      (1) r $\leftarrow$ r + 1

      (2) Select the first prototype in both set W^+ and W^-, and construct the closest prototype pair $R_r = (\omega_a, \omega_b)$ to be updated.

      (3) Update prototypes ω_a and ω_b based on Eq. (15) and (16).

(4) Update the matrix Ω according to Eq. (17). Ω is normalized to make sure that $\sum_i \Lambda_{ii} = 1$, in order to prevent the algorithm from degeneration [12, 13].

(5) Update learning rates η_ω and η_Ω based on Eq. (20) and (21).

(6) $W^+ \leftarrow W^+ \setminus \{\omega_a\}, W^- \leftarrow W^- \setminus \{\omega_b\}$

End While

End While

4 Experiments

Experiments were conducted on the provided real-life datasets. In this section, we introduce two different measures of evaluating the performance of the algorithms. All experimental settings were determined by cross-validation.

4.1 Evaluation

In the experiments, two evaluation methods were used to measure the accuracy of predicted results on a test set, namely, mean zero-one error (MZE) and mean absolute error (MAE). In the following, both measures are described in detail.

4.1.1 Mean zero-one error (MZE)

The mean zero-one error (MZE) is also known as the misclassification rate, reflecting the percentage of incorrect predictions among all the predicted results, without considering the class order. Formally, it is defined as

$$MZE = \frac{\sum_{i=1}^{n} I(y_i \neq \hat{y}(x_i))}{n}, \tag{22}$$

where n indicates the number of examples in the test set. The numerator $\sum I$ counts the number of misclassified points. The MZE value ranges from 0 to 1, which reveals a global performance for basic classification tasks without considering the class order. Hence, it is not preferred for ordinal classification or regression problems.

4.1.2 Mean absolute error (MAE)

The mean absolute error (MAE) measures the average absolute deviation of predicted ranks from true ranks with respect to the label space. It is given as

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - \hat{y}(x_i) \mid}{n}, \tag{23}$$

The MAE value ranges from 0 to $C-1$ (maximum rank difference). Since category information is included in this measurement, MAE is typically suitable for ordinal classification or regression tasks and can be used together with MZE [2].

4.2 Dataset

The performance of p-OGMLVQ was evaluated through real-world datasets by calculating the averages of MZE and MAE. Before tuning parameters for the algorithm, several preprocessing steps were conducted on the datasets, including normalization and upsampling.

4.2.1 Non-alcoholic fatty liver disease

The Non-alcoholic fatty liver disease (NAFLD) dataset used in the experiments contains steroid metabolomics data. NAFLD is a type of liver disorder caused by excess fat in the liver. It is the most common liver disorder in the western world as 20% to 30% of people in the west suffer from NAFLD [15]. The prevalence of NAFLD is expected to increase concurrence with the epidemic of obesity and diabetes mellitus [16].

The presence of excess fat in the liver can lead to nonalcoholic steatohepatitis (NASH), which has a high probability to progress to advanced liver disease. Once developed, 20% of the patients with NASH have symptoms of cirrhosis and 30% to 40% of these diagnosed patients suffer from liver-related death over a 10-year period [17]. Thus, early diagnosis and high accuracy of diagnosed stages are crucial.

Table 1: Distribution of Non-alcoholic Fatty Liver Disease (NAFLD) dataset

Rank	Count	Percentage
1	106	46.70%
2	16	7.05%
3	15	6.61%
4	8	3.52%
5	22	9.69%
6	60	26.43%

The ranking information in the NAFLD dataset describes different diagnosed stages of the disease. In total, there are 227 labelled examples with 33 features for each. Labels of all the instances range from 1 to 6, which indicate the severity of NAFLD in increasing order. The class distribution (percentage of examples in each class) is shown in Table 1. As can be seen, the dataset is severely unbalanced, where the percentage of data points per class ranges from 3.52% to 46.70%. Hence, the problem of unbalanced dataset needs to be handled before implementing the algorithms.

4.2.2 Preprocessing

Before implementing the p-OGMLVQ algorithm, prepossessing was performed on the entire dataset. Since the Euclidean distance was measured in the algorithm, all feature dimensions were normalized to have the properties of a standard normal

distribution (zero mean and unit variance), which makes sure that they contribute equally in the input space.

Table 2: Distribution of training and test set after 8-fold partition

Rank	Count	Training set	Test set
1	106	92	14
2	16	14	2
3	15	13	2
4	8	7	1
5	22	19	3
6	60	53	7

The dataset was then partitioned into training and test sets according to k-fold cross-validation. K-fold cross-validation is a widely used method for estimating prediction error. It uses part of the available data to fit a prediction model, and a different part to test the model [18].

For this dataset, we chose eight-fold cross-validation and roughly divided the data into eight equal-sized parts. For the kth part (k=1,2,...8), we fitted the model to other $k-1$ parts of the data and calculated the prediction error when testing on the kth part. We did this for $k = 1, 2, ...8$ and combined the eight estimates of the prediction error. In this case, $7/8$ of the data were used for training and $1/8$ for testing. The distribution of the training set and the test set for all classes can be seen from Table 2.

Table 3: Distribution of training and test set after upsampling procedure

Rank	Training set	Test set
1	92	14
2	92	14
3	92	14
4	92	14
5	92	14
6	92	14

It is worth mentioning that the two sets are still highly imbalanced with respect to their class distribution as recorded in Table 2. The number of data points in the training set ranges from 7 to 92, and that in the test set ranges from 1 to 14. Therefore, all classes were upsampled to have the same frequency in order to fairly contribute to the training model. However, in case upsampled data points reappear in the test set, which would yield over-optimistic classification results, we did the upsampling procedure separately for both training set and test set. The data points in each class were resampled to the maximum number of class examples with replacement. The final class distribution of the two sets is shown in Table 3. In the training set, there are 92 data points in each category, while in the test set,

14 testing examples exist in each category. Thus, there are 552 examples in the training set, and 84 test examples in the test set.

4.3 Experimental parameters

In this study, the performance of the p-OGMLVQ algorithm was evaluated against the standard nominal GMLVQ. All algorithm parameters were chosen through eight-fold cross-validation. Different combinations of candidate parameter sets were performed on the training set first, and the trained models were tested on the test set. Although we use two evaluation methods (MZE and MAE) in the experiments, MAE is the target in the selection of optimal parameters in the grid search as MAE reflects the category information and is typically suitable for ordinal classification problems.

For p-OGMLVQ, several parameters were tuned on the training sets through experiments. Number of prototypes per class ranged in $\{1, 2, 3, 4, 5\}$. The ranking loss threshold L_{min} were set among $\{0, 1, 2\}$. Candidate values for learning rates were listed as follows: $\eta_\omega \in \{0.07, 0.1, 0.15, 0.2\}$, $\eta_\Omega \in \{0.03, 0.08, 0.1, 0.15\}$. Gaussian kernel width were given among: $\sigma \in \{0.3, 0.5, 0.7\}$, $\sigma' \in \{0.7, 1.0, 1.5\}$. Thus, more than 100 combinations of these parameters were trained on the training set to fit the model. For GMLVQ, we implemented an online version of the original GMLVQ [19], as p-OGMLVQ updates the prototypes once randomly selecting a training example. In GMLVQ, the number of prototypes per class was tuned among $\{1, 2, 3, 4, 5\}$.

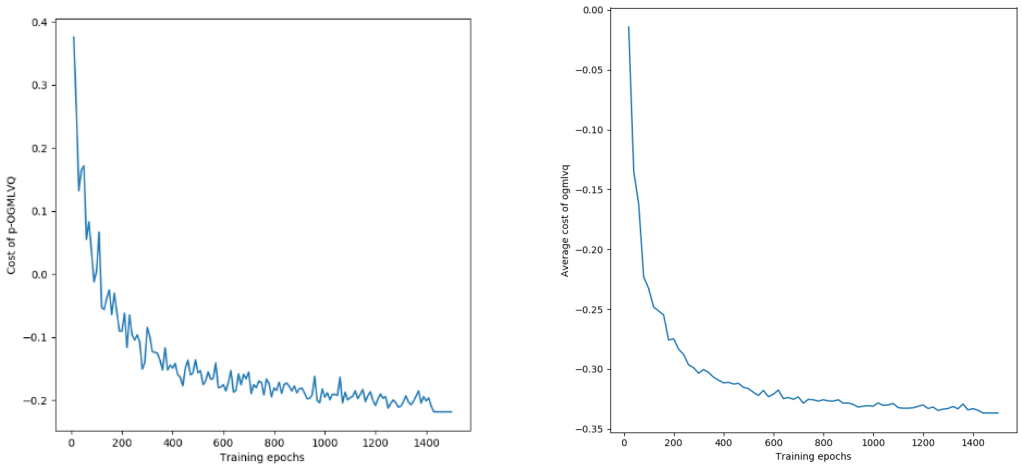
For both GMLVQ and p-OGMLVQ, experiments with tuned parameters were conducted 30 times. The performance of algorithms was evaluated based on the averages of MAE and MZE, together with their standard deviations.

5 Results

This section describes different results based on the implementation of the nominal GMLVQ algorithm and the ordinal GMLVQ (p-OGMLVQ) algorithm. We investigated the convergence behavior of their cost functions, as well as the average results of both MZE and MAE with increasing training epochs.

5.1 Cost function

Figure 2(a) shows the cost of the GMLVQ algorithm with growing training epochs. All data points were looped once over one training epoch. As given in Eq. (14), the overall cost function of p-OGMLVQ calculates the cost value based on all adapted prototype pairs. Since the number of adapted pairs varies during training phases, the average cost of each pair was calculated and traced as shown in Figure 2(b). Both cost values decrease with growing epochs and converge near the end of training epochs. However, p-OGMLVQ has a more stable convergence behavior.

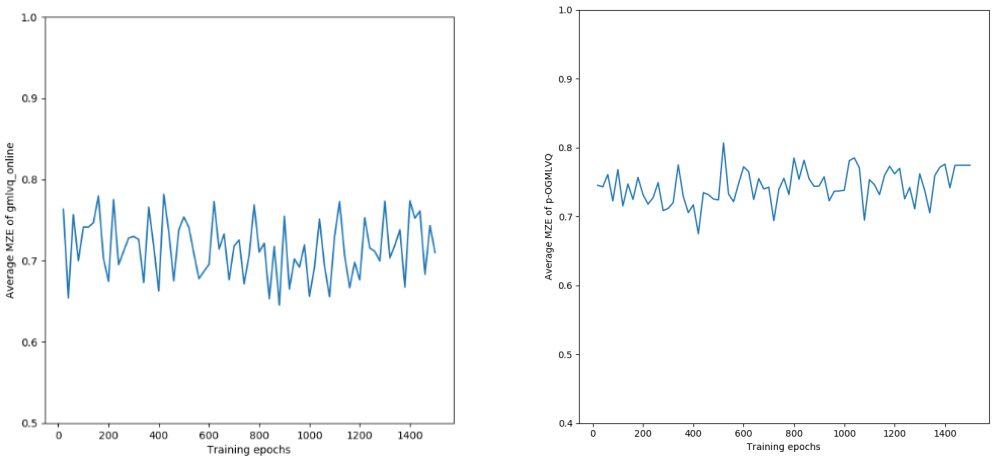


(a) Average cost of GMLVQ (b) Average cost of p-OGMLVQ

Figure 2: Average cost of each epoch

5.2 MZE and MAE

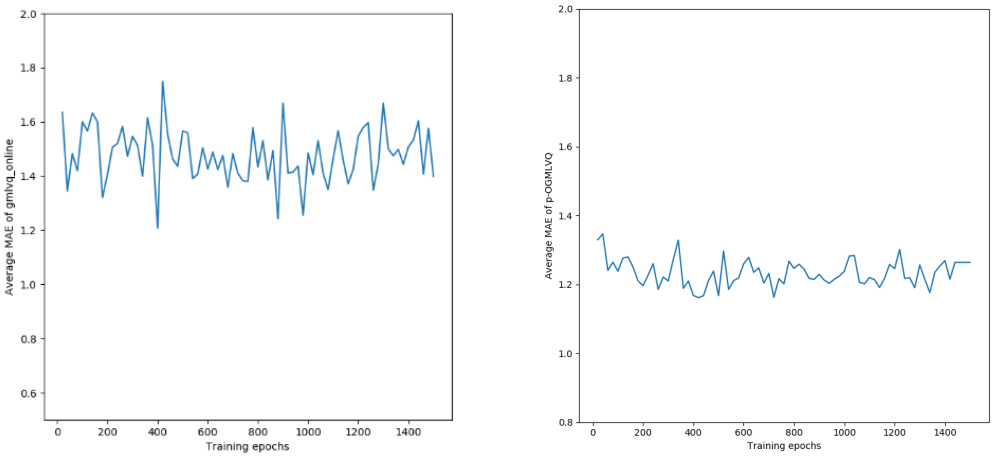
Both average MZE and MAE were calculated based on the experiments conducted across 30 runs using tuned parameters. As we did eight-fold cross-validation for one run, there were eventually 240 runs in total. Besides, the parameter values we used for p-OGMLVQ are listed as follows: the number of prototypes per class was set to 5; the ranking loss threshold was set to 1; the learning rates for prototypes and the metric were 0.07 and 0.3 respectively, and the Gaussian kernel width σ and σ' were set to 0.3 and 1. For GMLVQ, the number of prototypes per class was set the same as p-OGMLVQ.



(a) Average MZE of GMLVQ (b) Average MZE of p-OGMLVQ

Figure 3: Average mean zero-one error (MZE)

Figure 3 and 4 represent the changes of MZE and MAE on test sets with increasing training epochs. Both proposed p-OGMLVQ and original GMLVQ method show similar patterns with respect to MZE in Figure 3, while GMLVQ slightly outperforms with a lower average value of MZE around 0.71 and p-OGMLVQ achieves approximately 0.78. However, for ordinal classification tasks, the standard evaluation measure (MZE) is insufficient due to the lack of order information. Therefore, along with MZE, MAE was evaluated during the experiments, which is typically used for ordinal classification.



(a) Average MAE of GMLVQ (b) Average MAE of p-OGMLVQ

Figure 4: Average mean absolute error (MAE)

The results of the mean absolute error (MAE) evaluated on the test sets are demonstrated in Figure 4. Compared with the original GMLVQ in Figure 4(a), p-OGMLVQ has distinct advantages with a lower and more stable value of average MAE around 1.25 as shown in Figure 4(b). On the other hand, the MAE of GMLVQ shows larger jumps approximately ranging from 1.2 to 1.8.

Table 4: MZE and MAE on test sets across 30 runs

	MZE		MAE	
	Mean	SD	Mean	SD
GMLVQ	0.710	0.085	1.426	0.289
p-OGMLVQ	0.787	0.031	1.251	0.037

The average MZE and MAE were calculated along with their standard deviations from the last training epoch, as shown in Table 4. The mean values describe the overall performance of algorithms, while the standard deviations reveal the fluctuations concerning the test behavior. As expected, the p-OGMLVQ approach achieves better performance in MAE reaching 1.251 ± 0.037, while the original GMLVQ model obtains 1.426 ± 0.289.

6 Discussion

As an extended version of GMLVQ, p-OGMLVQ takes order information into consideration. It shows improvements over nominal algorithms in terms of MAE instead of MZE. In other words, more misclassifications happen within a tolerant range owing to the tuned ranking loss threshold, which yields a smaller absolute error but a larger mean-zero error. Thus, p-OGMLVQ avoids misclassifications with large rank differences and accepts misclassifications within an acceptable range. It makes sense in many real-world classification problems since it is usually more severe to misclassify a class with larger rank differences, for example, when judging illness conditions in the medical field. In the nominal GMLVQ, the absolute error fluctuates wildly without taking class orders into account.

6.1 Impacts of parameters

In the p-OGMLVQ algorithm, many parameters need to be tuned during the training phase, such as the number of prototypes per class, learning rates, the ranking loss threshold and the Gaussian kernel bandwidth. All these parameters are important and have different impacts on the performance of the algorithm. In the experiments, we find that better MAE results can always be achieved if there are more prototypes exist in each class. For the learning rates η_ω and η_Ω, both are determined by their initial values together with the other two parameters, the decay factor and training epochs. Thus, the initial settings of the decay factor and training epochs are also important.

Besides, the Gaussian kernel width σ and σ' determine the weights for correct and incorrect prototypes respectively. For correct prototypes, a large ranking difference results in a small weight. Conversely, a small ranking difference has a large weight. For incorrect prototypes, however, weights are more complicated as the weight factor is determined by two terms in Eq. (12). When the incorrect prototype is spatially closer to the input example and has a label further from that of the input example, both two terms become larger and lead to a larger weight. Thus, the σ' in the denominator of the second term cannot be too small, which may result in an opposite result.

Moreover, the ranking loss threshold L_{min} was set to 1 in the end, as both 0 and 2 did not show improvements over GMLVQ. When L_{min} equals to 0, there is no tolerable ranking difference, and only one prototype pair will be selected to update, which is the same as GMLVQ. When L_{min} equals to 2, there can be a maximum of five correct classes, but the dataset only has six rankings, which is also not preferred for the classification.

6.2 Limitations and potential problems

In this study, there are some limitations and potential problems that hinder the performance of the proposed algorithm.

At first, the dataset plays a vital role in the performance of the algorithms as it contains all the feature vectors. However, as can be seen in Table 1, the medical liver dataset is severely imbalanced. Class 1 takes up 46.7%, almost half of the entire examples, while class 4 only accounts for 3.52%. Even though the upsampling procedure was conducted in the experiments, insufficient information for classes with fewer data points would result in weak performance. Besides, the diagnosed ranking information in the dataset may have noise and outliers could exist in the dataset as well, which can also have negative impacts on the classification results.

In addition, as the learning rate controls the step size of the algorithm, it has to be small enough to ensure convergence, but also should be as large as possible to facilitate fast learning. For the cost function, the aim is to find its global optimality, but there can be many local minima that are far away from it. Thus, a decay factor is used to control the annealing speed. It has to be slow enough to leave local minima, but also fast enough to achieve solutions in realistic computing time. Besides, the initial conditions of the learning rates are also of great importance as they determine which minimum will be approached.

Furthermore, the cost function of p-OGMLVQ is given as the sum of r weighted versions of the GMLVQ cost function, but the number of adapted prototype pairs varies during the training phase, which cannot guarantee the monotonic decreasing of the cost function. Thus, we calculated the average cost of all pairs. Besides, as mentioned in [3], the cost function of p-OGMLVQ does not treat correct/incorrect prototypes equally due to the changing values of the denominator. This strategy breaks the global class order, which could make a less correct/incorrect prototype update more and cannot consistently reflect global ordering relations among prototypes. Thus, an accumulated version of the cost has been proposed by [3].

When it comes to the weights for incorrect prototypes, we find that the Gaussian kernel width σ' cannot be too small or too big. If σ' is too small, for a small distance between the wrong prototype and the input instance, the weight will be close to zero, which should be large for updating the wrong prototype. Conversely, if σ' is too big, it is not suitable for a large distance. Hence, it can be challenging to find an optimal σ' that is suitable for all distances. It could be better if both Gaussian kernel width σ and σ' can be self-adapted during the training phase so that there would be no extra effort on finding the optimal values manually in the experiments.

As we mentioned previously, there are more than 100 combinations of different parameters in the experiments, and more candidate parameter values are needed if we want to achieve more precise results. Therefore, it is not only computationally expensive but also makes it difficult to find the best parameter set because the average MAE results do not show apparent differences among all these parameter sets.

7 Conclusion

In the study, the pair ordinal GMLVQ (p-OGMLVQ) was implemented as an extended realization of GMLVQ [12, 13], which belongs to the family of prototype-based supervised learning algorithms in the field of machine learning.

In p-OGMLVQ, ranking information is particularly taken into account during the training phase. The ordinal information is used to construct the prototype pairs to be adapted and determine the degree of updating by two weighting factors. A ranking loss threshold is also set to obtain a tolerable range of ranking error. With the acceptable ranking loss, MAE is mainly used for evaluating the ordinal classification performance. MZE may perform worse in this case. For example, if the original class of an input example is 2, the predicted class 3 and 4 are equally treated as misclassification in terms of MZE. However, MAE counts the deviation from the correct rank, which takes the class order into account. Therefore, compared to GMLVQ, p-OGMLVQ outperforms with respect to MAE not only in the NAFLD dataset but also in other datasets [1].

One difficulty of this study is the imbalanced dataset with limited feature information. We focused on the upsampling procedure together with k-fold cross-validation, in order to avoid trained data points reappear in the test set, which would result in over-optimistic outcomes. Besides, the dataset may have some diagnosed noise or outliers. Thus, it could be better if we do more exploratory analysis and data cleansing on the dataset in the future, such as finding unusual values, making corrections of data points and removing outliers.

Another problem is that there are many parameters in the p-OGMLVQ algorithm to be tuned in the experiments. Thus, it can be quite computationally expensive comparing with GMLVQ when finding the optimal parameter set. If some of the parameters can be self-adapted in the training phase, as the update rules of the Gaussian kernel width presented in [3], it could save computational effort and also obtain a more precise result.

In the future, ROC curves [20] can be produced as well, together with MZE/-MAE measurements, as the non-alcoholic fatty liver disease (NAFLD) dataset is a medical dataset. In addition, other data sets can also be used for comparing the performance between p-OGMLVQ and the standard GMLVQ.

References

[1] S. Fouad and P. Tiňo. Adaptive metric learning vector quantization for ordinal classification. *Neural Computation*, 24(11):2825–2851, 2012.

[2] J. Sánchez-Monedero, Pedro A. Gutiérrez, Peter Tiňo, and C. Hervás-Martínez. Exploitation of pairwise class distances for ordinal classification. *Neural Comput.*, 25(9):2450–2485, September 2013.

[3] Fengzhen Tang and Peter Tiňo. Ordinal regression based on learning vector quantization. *Neural Networks*, 93:76 – 88, 2017.

[4] Harmen J Dikkers. Support vector machines in ordinal classification. 09 2005.

[5] Bob Armstrong and M. A. Sloan. Ordinal regression models for epidemiologic data. *American journal of epidemiology*, 129 1:191–204, 1989.

[6] Irina Yatskiv and Nadezda Kolmakova. Using ordinal regression model to analyze quality of service for passenger terminal. 05 2011.

[7] Teuvo Kohonen. Learning vector quantization for pattern recognition. Report TKK-F-A601, Laboratory of Computer and Information Science, Department of Technical Physics, Helsinki University of Technology, Helsinki, Finland, 1986.

[8] Teuvo Kohonen. *Learning Vector Quantization.* Self-Organizing Maps, 1995.

[9] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[10] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 423–429, Cambridge, MA, USA, 1995. MIT Press.

[11] Barbara Hammer and Thomas Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8):1059 – 1068, 2002.

[12] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Comput.*, 21(12), December 2009.

[13] Petra Schneider. *Advanced methods for prototype-based classification.* PhD thesis, 2010. Relation: https://www.rug.nl/ Rights: University of Groningen.

[14] Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. Ordinal classification with decision rules. In *Proceedings of the Third International Conference on Mining Complex Data*, pages 169–181, Berlin, Heidelberg, 2007. Springer-Verlag.

[15] Michael H. Le, Pardha Devaki, Nghiem B. Ha, Dae Won Jun, Helen S. Te, Ramsey C. Cheung, and Mindie H. Nguyen. Prevalence of non-alcoholic fatty liver disease and risk factors for advanced fibrosis and mortality in the united states. *PLOS ONE*, 12(3):1–13, 03 2017.

[16] Jacob C. Seidell. Obesity, insulin resistance and diabetes — a worldwide epidemic. *British Journal of Nutrition*, 83(S1):S5–S8, 2000.

[17] Arthur Mccullough. The clinical features, diagnosis and natural history of nonalcoholic fatty liver disease. 8:521–33, viii, 09 2004.

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. 02 2009.

[19] Petra Schneider, Frank-Michael Schleif, Thomas Villmann, and Michael Biehl. Generalized matrix learning vector quantizer for the analysis of spectral data. In *ESANN*, 2008.

[20] J.A. Hanley and Barbara Mcneil. The meaning and use of the area under a receiver operating characteristic (roc) curve. 143:29–36, 05 1982.