LGM-MCE Training Based on Parzen Estimation with Sample Weighting

David Ha[†] Hideyuki Watanabe[‡] Shigeru Katagiri[†] Miho Ohsaki[†]

1. Introduction

Large Geometric Margin Minimum Classification Error (LGM-MCE) training aims to enhance the robustness of trained classifiers to unseen samples through the joint execution of classification error minimization and geometric margin maximization [1]. To do so, it needs to appropriately control the smoothness of its classification error count loss.

Conventionally, an optimal smoothness setting is searched for empirically based on many experimental training runs, each using a different degree of smoothness. However, such a heuristic setting does not guarantee that a resulting setting is optimal. To solve this problem, methods based on the Parzen estimation of the Bayes risk have been proposed [2, 3]. In these methods, smoothness was automatically determined by the Parzen window width used in the Parzen estimation that was performed in the geometric misclassification measure space [*]

The Parzen-estimation-based methods successfully superseded the conventional empirical way. However, because they used a uniform window width value for all the Parzen windows, their resulting smoothness was not always sufficiently optimized. Therefore, since there is room for further improvement in the smoothness estimation, we propose in this paper a new reformalization of LGM-MCE training using Parzen estimation with sample weighting [4], where each window width is adapted based on the sample density in the geometric misclassification measure space. In the following sections, we detail our new formalization and report its experimental evaluation results.

2. Large Geometric Margin Minimum Classification Error Training

2.1 Geometric Misclassification Measure

We consider the task of classifying input sample $\boldsymbol{x}$, extracted from infinite feature space $\mathcal{X}$, into one of the J classes $(C_1, \ldots, C_J)$. For the convenience of subsequent discussions, we assume $\boldsymbol{x}$ is of fixed dimension, although LGM-MCE training can also handle variable-length patterns. Classification process $C(\,)$ takes the following general form:

$$C(\boldsymbol{x}) = C_k \;\text{ iff }\; k = \arg\max_j g_j(\boldsymbol{x};\Lambda), \tag{1}$$

where Λ is a set of classifier parameters (class models) to be trained and $g_j(\boldsymbol{x};\Lambda)$ is a discriminant function of C_j, which is differentiable in Λ. The value of $g_j(\boldsymbol{x};\Lambda)$ represents the degree to which $\boldsymbol{x}$ belongs to C_j.

[†]Doshisha University, [‡]ATR

[*]For presentation purposes, we refer to the misclassification measure defined using geometric margin as geometric misclassification measure.

To represent the classification result, LGM-MCE training computes the following geometric misclassification measure for training input $\boldsymbol{x}$ belonging to C_y:

$$D_y(\boldsymbol{x};\Lambda) = \frac{d_y(\boldsymbol{x};\Lambda)}{\|\nabla_{\boldsymbol{x}} d_y(\boldsymbol{x};\Lambda)\|}, \tag{2}$$

where $\boldsymbol{x}^{\dagger}$ is the closest training sample to the class boundary and $d_y(\boldsymbol{x};\Lambda)$ is a misclassification measure, usually defined as

$$d_y(\boldsymbol{x};\Lambda) = -g_y(\boldsymbol{x};\Lambda) + \max_{j, j\neq y} g_j(\boldsymbol{x};\Lambda). \tag{3}$$

$D_y(\boldsymbol{x};\Lambda)$ represents the classification result for $\boldsymbol{x}$ using Λ. When the value of $D_y(\boldsymbol{x};\Lambda)$ is negative, its corresponding classification is correct and incorrect for the positive value. $D_y(\boldsymbol{x};\Lambda)$ is also a sign-reversed version of the geometric margin that approximates the geometric (Euclidian) distance[†] between input $\boldsymbol{x}$ and its closest class boundary. Maximizing the geometric margin increases the robustness to unseen samples [5]. Note here that a geometric margin value observed in the (single-dimension) geometric misclassification measure space equals that in the (usually high-dimension) sample space. This relation is illustrated in Fig. 1, where r expresses the geometric margin. LGM-MCE training increases the absolute measure values (correct classification results) for the training inputs in the negative region of the geometric misclassification measure space and thus maximizes geometric margin value r. We elucidate this mechanism in the next subsection.

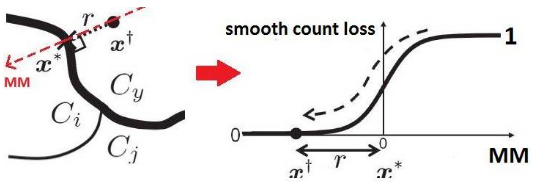

Figure 1: Sigmoidal smooth error count loss

2.2 Loss

For each training sample $\boldsymbol{x}$, LGM-MCE training maps a classification result observed in the geometric misclassification measure space to a score whose value is between 0 and 1, which approximates the classification error count. A typical mapping function is the following sigmoidal loss (right side of Fig. 1):

$$\ell_y((D_y(\boldsymbol{x};\Lambda)) = \frac{1}{1+\exp(-\alpha_y(D_y(\boldsymbol{x};\Lambda))}, \tag{4}$$

[†]When $g_j(\boldsymbol{x};\Lambda)$ is the linear discriminant function and the (sign-reversed) prototype-based distance, $D_y(\boldsymbol{x};\Lambda)$ is equal to the geometric margin without approximation.

where α is a positive constant that expresses the loss smoothness. As α takes larger values, the loss tends to the 0-1 ideal classification error count. The classification correctness (resp. incorrectness) increases as the score gets closer to 0 (resp. 1).

The above individual loss scores are then combined into the following penalty function that is called empirical average loss $L(\Lambda)$, which is minimized to reduce the classification errors over the entire set of training samples and pursue the desirable status of classifier parameters Λ:

$$L(\Lambda) = \frac{1}{N}\sum_{n=1}^{N} \ell_{y_n}(D_{y_n}(\boldsymbol{x}_n;\Lambda)). \tag{5}$$

The smoothness of $L(\Lambda)$ in Λ enables such minimization procedures as the probabilistic descent method [6], the steepest descent method, or Rprop [7].

2.3 Automatic determination of loss smoothness

The mechanism of increasing the geometric margin through the minimization of the smooth classification error count loss requires hyperparameter α to be determined to accelerate the geometric margin maximization as well as the classification error minimization. Conventionally, this determination was empirically conducted based on many training trials. Clearly, such an experimental process is time-consuming and does not guarantee an optimal determination, which leads to the accurate estimation of the Bayes risk, or in other words, high training robustness to unseen samples.

To solve this problem, a method separately estimating α_y for each class C_y was proposed by reformalizing LGM-MCE training using Parzen estimation in geometric misclassification measure space [2]. The reformalization starts by rewriting the expected loss (Bayes risk) $R(\Lambda)$ in the geometric misclassification measure space:

$$\begin{aligned} R(\Lambda) &= \sum_{y=1}^{J} P(C_y)\int_{\mathcal{X}} 1\left(D_y(\boldsymbol{x};\Lambda) > 0\right) p(\boldsymbol{x}|C_y)\mathrm{d}\boldsymbol{x} \\ &= \sum_{y=1}^{J} P(C_y)\int_{0}^{\infty} p(z|C_y)\mathrm{d}z, \end{aligned} \tag{6}$$

where z is a sample point in the misclassification measure space. Eq. (6) suggests that the expected loss, which was originally defined in a high-dimension sample space, can be represented by estimating conditional probability density $p(z|C_y)$ in one-dimensional misclassification measure space. For example, $p(z|C_y)$ can be estimated by the following Parzen estimate:

$$\hat{p}(z|C_y;h_y) = \frac{1}{N_y}\sum_{n=1}^{N_y}\frac{1}{h_y}\phi\Big(\frac{z - D_y(\boldsymbol{x}_n^y;\Lambda)}{h_y}\Big), \tag{7}$$

where $\boldsymbol{x}_n^y$ denotes the n-th training sample belonging to C_y, N_y denotes the total number of training samples belonging to C_y ($N = \sum_{y=1}^{J} N_y$), and $\frac{1}{h_y}\phi\Big(\frac{z-D_y(\boldsymbol{x}_n^y;\Lambda)}{h_y}\Big)$ represents a Parzen window whose width h_y is centered on $D_y(\boldsymbol{x}_n^y;\Lambda)$ in the geometric misclassification measure space.

Substituting the density estimate in Eq. (7) to the true density in (6) leads to a finite-sample-based estimate of $R(\Lambda)$, i.e., $R_N(\Lambda)$:

$$R_N(\Lambda) = \frac{1}{N}\sum_{y=1}^{J}\sum_{n=1}^{N_y}\int_0^{\infty}\frac{1}{h_y}\phi\Big(\frac{z - D_y(\boldsymbol{x}_n^y;\Lambda)}{h_y}\Big)dz, \tag{8}$$

where $P(C_y)$ is approximated by N_y/N. Approximating in turn $R_N(\Lambda)$ in (8) with $L(\Lambda)$ in (5) redefines the smooth error count loss (4) as an integral defined in the misclassification measure space.

$$\ell_y(D_y(\boldsymbol{x};\Lambda)) = \int_0^{\infty}\frac{1}{h_y}\phi\Big(\frac{z - D_y(\mathbf{x}_n^y;\Lambda)}{h_y}\Big)dz. \tag{9}$$

Following the above preparations, an optimal value of h_y is estimated with the risk fs maximum likelihood estimation. By applying the Cross-Validation Maximum Likelihood (CVML) method, the following likelihood function is defined:

$$f(h_y) = \prod_{n=1}^{N_y}\Big\{\frac{\sum_{m\neq n}^{N_y}\frac{1}{h_y}\phi\Big(\frac{D_y(\mathbf{x}_n^y;\Lambda)-D_y(\mathbf{x}_m^y;\Lambda)}{h_y}\Big)}{N_y - 1}\Big\}. \tag{10}$$

Maximizing $f(h_y)$ in terms of h_y leads to an optimal value of h_y, which is expected to correspond to an accurate estimation of the Bayes risk.

Among many, the Gaussian window is a natural and handy selection. In this case, identifying (9) with the original definition (4) leads to the following relation between Parzen window width h_y and loss smoothness parameter α_y [8]:

$$\alpha_y = \frac{4}{\sqrt{2\pi}h_y}. \tag{11}$$

Accordingly, substituting the h_y value gained through the maximization of (10) to (11) provides an optimal setting of the loss smoothness.

3. Variable Parzen Window Width Estimation of Loss Smoothness

Usually, the dataset contains more training samples in the center of the sample distribution (often far from the class boundaries), while there are fewer samples near the class boundaries. This nature of sample distribution affects the estimation of h_y. When the Parzen window adopts a uniform window width, the estimated width tends to be heavily influenced by the (many) training samples in the region far from the class boundaries and is probably unsuited for achieving high robustness. To solve this problem, we adopt Parzen estimation using variable window widths and assign wider windows (smoother loss) to the regions where the sample distribution is sparse and set narrower windows (sharper loss) to the regions where the samples are densely located.

3.1 Parzen Estimation with Sample-Dependent Weighting

The Parzen estimate with variable window widths in the original sample space can be formally written [10]:

$$\hat{p}(\boldsymbol{x}; h_1, ..., h_N) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h_n}\phi\Big(\frac{\boldsymbol{x}-\boldsymbol{x}_n}{h_n}\Big), \qquad (12)$$

where h_n denotes the window width for sample $\boldsymbol{x}_n$. Following the automatic loss smoothness determination procedure in 2.3, we rewrite (12) for the estimated density defined in the geometric misclassification measure space in the class-by-class mode:

$$\hat{p_y}(z; h_1^y, ..., h_{N_y}^y) = \frac{1}{N_y}\sum_{n=1}^{N_y}\frac{1}{h_n^y}\phi\Big(\frac{z-z_n^y}{h_n^y}\Big), \qquad (13)$$

where $\boldsymbol{x}^y$ denotes a sample of C_y, h_n^y denotes the window width for the n-th sample belonging to C_y, $z = D_y(\boldsymbol{x}^y; \Lambda)$, and $z_n^y = D_y(\boldsymbol{x}_n^y; \Lambda)$.

A straightforward way of estimating the sample-dependent window widths $\{h_1^y, \cdots, h_{N_y}^y\}$ is to apply the maximum likelihood estimation, as in 2.3. However, this step is obviously undesirable. Unfortunately, performing the maximum likelihood estimation in regard to N_y variables, $\{h_1^y, \cdots, h_{N_y}^y\}$, over N_y samples results in less robust estimation. To avoid this problem, we perform the following decomposition into common factor h_y and sample-dependent contribution w_n^y to preserve class-by-class formalization through class common factor h_y while producing sample-dependent Parzen windows $\{h_n\}_{n=1}^{N_y}$:

$$h_n^y = h_y w_n^y \qquad (14)$$

Algorithm 1 introduces a practical procedure for weights estimation, assuming that common factor h_y is adequately re-estimated at each step. In the algorithm, the geometric mean (line 3) merely acts as a normalization factor so that the product of the sample weights amounts to 1.

3.2 Cross Validation Maximum Likelihood Estimation of Common Factor h_y

In this subsection, we detail line 4 in Algorithm 1, which corresponds to the estimation of common factor h_y based on a fixed set of widths $\{w_i^y\}^{(k)}$. Following the procedure in 2.3, we adopt the CVML estimation method, which consists of the following two steps: maximum likelihood and cross validation.

In the maximum likelihood estimation step, we need to define a likelihood function over the geometric misclassification measure space as an objective function for maximization. According to (6), its maximization improves the quality of the risk estimation in regard to which parameters Λ should be optimized.

```
  Input: $k = 0, {h_y}^{(0)}; \forall n \in [1, N_y], {w_n}^{(0)} = 1$
  Output: ${h_y}^{(k+1)}, \{w_i^y\}_{i=[1,N_y]}^{(k+1)}$
1 while convergence condition not reached do
      /* Step 1: Computations of weights $\{w_i^y\}^{(k+1)}$ */
2     $\forall n \in [1, N_y], {w_n^y}^{(k+1)} = \left(\frac{\hat{p}(x_n^y, \{h_i^y\}^{(k)})}{g}\right)^{-\eta}$   (15)
3     where $g = \left\{\prod_{n=1}^{N_y} \hat{p}\left(x_n^y, \{h_i^y\}^{(k)}\right)\right\}^{\frac{1}{N_y}}$
      // previous studies [4] recommend $\eta = 0.5$ to reduce the bias in (13)
      /* Step 2: Estimation of common factor $h_y$ */
4     Estimate ${h_y}^{(k+1)}$ based on $\{w_i^y\}^{(k+1)}$   // cf. Algorithm 2
      /* Parzen widths update */
5     $\forall n \in [1, N_y], {h_n^y}^{(k+1)} \leftarrow {h_y}^{(k+1)} {w_n^y}^{(k+1)}$
6     $k \leftarrow k + 1$
7 end
```

Algorithm 1: Sample-weighted Parzen width estimation relative to class y

The step starts by defining a density estimate on the entire training set:

$$\hat{p}(z|C_y; h_y, \{w_i^y\}) = \frac{1}{N_y}\sum_{m=1}^{N_y}\frac{1}{h_y w_m^y}\phi\Big(\frac{z-z_m^y}{h_y w_m^y}\Big). \quad (16)$$

However, this estimate simply returns an optimal but meaningless value $h_y = 0$ when it is used in the maximum likelihood estimation [11].

To avoid this problem, we redefine the density estimate for the n-th transformed data point z_n^y $(= D_y(\boldsymbol{x}_n^y; \Lambda))$ by removing z_n^y itself from given N_y data points $\{z_m^y\}_{m\in[1,N_y]}$:

$$\hat{p}_{-n}(z|C_y; h_y, \{w_i^y\}) = \frac{1}{N_y - 1}\sum_{m\neq n}^{N_y}\frac{1}{h_y w_m^y}\phi\Big(\frac{z-z_m^y}{h_y w_m^y}\Big). \quad (17)$$

As an objective function of h_y to maximize, we next define the likelihood function that is the product of the density estimates over $\{z_n^y\}_{n\in[1,N_y]}$:

$$f(h_y; \{w_i^y\}) = \prod_{n=1}^{N_y} \hat{p}_{-n}\left(z_n^y|C_y; h_y, \{w_i^y\}\right) \qquad (18)$$

The maximization (18) improves the quality of the risk estimation (in regard to which parameters Λ should be optimized) in the geometric misclassification measure space. Although such gradient descent methods as the steepest descent method are possible ways of maximizing $f(h_y; \{w_i^y\})$, they require careful manual initialization of the learning coefficient. To alleviate this problem, we reformalize the CVML estimation procedure using the concept of auxiliary function. First, we rewrite $f(h_y; \{w_i^y\})$ in a more optimization-friendly form:

$$F_y(h_y; \{w_i^y\}) = \sum_{n=1}^{N_y} \ln \left\{ \sum_{m \neq n}^{N_y} \frac{1}{h_y w_m^y} \phi\left(\frac{z_n^y - z_m^y}{h_y w_m^y} \right) \right\}. \tag{19}$$

Since Eq. (19) is a monotonically increasing function of (18), optimizing either (19) or (18) is equivalent. Here we define

$$q_{n,m}^y(h_y; \{w_i^y\}) = \frac{\frac{1}{h_y w_m^y} \phi\left(\frac{z_n^y - z_m^y}{h_y w_m^y} \right)}{\sum_{k \neq n}^{N_y} \frac{1}{h_y w_k^y} \phi\left(\frac{z_n^y - z_k^y}{h_y w_k^y} \right)}, \tag{20}$$

where $\{q_{n,m}^y\}_{m \neq n}^{N_y}$ satisfies $q_{n,m}^y > 0$ and $\sum_{m \neq n}^{N_y} q_{n,m}^y = 1$. Assuming that $F_y(h_y; \{w_i^y\})$ is optimized by an iterative procedure and that estimate $\hat{h_y}$ has already been calculated before the last preceding iteration step, we define the following auxiliary function for successive iteration steps:

$$\begin{aligned} W_y(h_y; \{w_i^y\}) &= \sum_{n=1}^{N_y} \sum_{m \neq n}^{N_y} q_{n,m}^y(\hat{h_y}; \{w_i^y\}) \\ &\times \ln \left\{ \frac{\frac{1}{h_y w_m^y} \phi\left(\frac{z_n^y - z_m^y}{h_y w_m^y} \right)}{q_{n,m}^y(\hat{h_y}; \{w_i^y\})} \right\} \end{aligned} \tag{21}$$

We also define the following difference function:

$$K_y(h_y; \{w_i^y\}) = F_y(h_y; \{w_i^y\}) - W_y(h_y; \{w_i^y\}). \tag{22}$$

Substituting (19) and (21) into (22) and considering (20), we reach the following expression:

$$\begin{aligned} &K(h_y; \{w_i^y\}) \\ &= -\sum_{n=1}^{N_y} \sum_{m \neq n}^{N_y} q_{n,m}^y(\hat{h_y}; \{w_i^y\}) \ln \{\mathcal{Q}\} \\ &\geq \underbrace{\sum_{n=1}^{N_y} \sum_{m \neq n}^{N_y} q_{n,m}^y(\hat{h_y}; \{w_i^y\}) \{1 - \mathcal{Q}\}}_{0}, \end{aligned} \tag{23}$$

where $\mathcal{Q} = q_{n,m}^y(h_y; \{w_i^y\}) / q_{n,m}^y(\hat{h_y}; \{w_i^y\})$. In (23), we used logarithm-based inequality $\forall x > 0, -\ln(x) \geq 1 - x$, which only achieves equality for $x = 1$. Therefore, the inequality in (23) becomes an equality if and only if $\forall n, m \in [1, N_y], m \neq n, q_{n,m}^y(h_y) = q_{n,m}^y(\hat{h_y})$. In other words, $K_y(h_y; \{w_i^y\})$ is minimized at $h_y = \hat{h_y}$, and the minimum value is zero. Considering (22), this leads to:

$$\begin{cases} F_y(\hat{h_y}; \{w_i^y\}) = W_y(\hat{h_y}; \{w_i^y\}), & (24) \\ \nabla_{h_y} F_y(\hat{h_y}; \{w_i^y\}) = \nabla_{h_y} W_y(\hat{h_y}; \{w_i^y\}). & (25) \end{cases}$$

It can be seen from (23) and (24) that $F_y(h_y; \{w_i^y\})$ exceeds $F_y(\hat{h_y}; \{w_i^y\})$ when $W_y(h_y; \{w_i^y\})$ exceeds $W_y(\hat{h_y}; \{w_i^y\})$ based on the following expression:

$$\begin{aligned} &F_y(h_y; \{w_i^y\}) \geq W_y(h_y; \{w_i^y\}) > \\ &\qquad \left(W_y(\hat{h_y}; \{w_i^y\}) = F_y(\hat{h_y}; \{w_i^y\}) \right). \end{aligned}$$

Furthermore, based on (25), unless $\hat{h_y}$ is a stationary point of $F_y(h_y; \{w_i^y\})$, gradient $\nabla_{h_y} W_y(\hat{h_y}; w_1^y, ..., w_n^y)$ is nonzero, and we can find h_y such that $W_y(h_y; \{w_i^y\}) > W_y(\hat{h_y}; \{w_i^y\})$. Consequently, by finding h_y such that $W_y(h_y; \{w_i^y\}) > W_y(\hat{h_y}; \{w_i^y\})$,replacing $\hat{h_y}$ with h_y, and repeating this process with a proper initial value of $\hat{h_y}$, $F_y(h_y; \{w_i^y\})$ monotonically increases until h_y reaches its local maximum point.

Consequently, point h_y, which maximizes auxiliary function $W_y(h_y; \{w_i^y\})$, is given as the following closed-form formula:

$$h_y = \sqrt{\frac{1}{N_y} \sum_{n=1}^{N_y} \sum_{m \neq n}^{N_y} q_{n,m}^y(\hat{h_y}; \{w_i^y\}) \left(z_n^y - z_m^y \right)^2}. \tag{26}$$

Using the converged value of h_y, we achieve a desired value of loss smoothness hyperparameter α_y by (11) in the newly-formalized LGM-MCE framework described in Section 3.3.

In the case of a Gaussian kernel, $q_{n,m}^y(h_y; \{w_i^y\})$ takes the form shown in (28), and the likelihood function becomes:

$$\begin{aligned} F(h_y; \{w_i^y\}) &= \sum_{y=1}^{J} F_y(h_y; \{w_i^y\}) = \\ &\sum_{y=1}^{J} \sum_{n=1}^{N_y} \ln \left(\sum_{m \neq n}^{N_y} \frac{1}{w_m^y} \exp\left\{ -\frac{1}{2} \left(\frac{z_n^y - z_m^y}{h_y w_m^y} \right)^2 \right\} \right) \\ &- \sum_{y=1}^{J} N_y \ln h_y. \end{aligned} \tag{27}$$

The implementation of the above CVML estimation is summarized in Algorithm 2, which is in the Expectation-Maximization (EM) optimization form. To start the algorithm effectively, we must appropriately initialize h_y. Variable $q_{n,m}^y$ corresponds to the degree to which each data point z_n^y is assigned to another point $z_m^y (m \neq n)$. Considering this, we can assume the following possible initialization for this variable by redefining $q_{n,m}^y(h_y)$:

$$q_{n,m}^y(h_y) = \begin{cases} 1, & \text{if } m = k(n), \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

In other words, each data point z_n^y is assigned to its nearest-neighbor $z_{k(n)}^y$. Substituting (30) into (29)

```
Input: $l \leftarrow 0, {h_y}^{(0)}$
;                                   /* cf. Equation 31 */
Output: Optimized window width ${h_y}^{(l+1)}$ relative
        to class $y$
1 while ${h_y}^{(l+1)}$ does not meet convergence condition
  do
2     (Expectation step) Compute:
      $$q^y_{n,m}(h_y;\{w^y_i\}) = \frac{\frac{1}{w_m}\phi\left(\frac{z^y_n - z^y_m}{h_y w^y_m}\right)}{\sum_{k\neq n}^{N_y}\frac{1}{w_k}\phi\left(\frac{z^y_n - z^y_k}{h_y w^y_m}\right)} \quad (28)$$
      (Maximization step) Optimized Parzen width
      $${h_y}^{(l+1)} = \sqrt{\frac{1}{N_y}\sum_{n=1}^{N_y}\sum_{m\neq n}^{N_y} q^y_{n,m}(\hat{h_y};\{w^y_i\})\mathcal{Z}^2} \quad (29)$$
3     $l \leftarrow l+1$
4 end
```

Algorithm 2: Class-by-class Cross Validation Maximum Likelihood estimation of common factor h_y relative to class y. Here, $\mathcal{Z} = \left(\frac{z^y_n - z^y_m}{w^y_m}\right)$.

gives the following initial value for h_y:

$$ {h_y}^{(0)} = \sqrt{\frac{1}{N_y}\sum_{n=1}^{N_y}\left(\frac{z^y_n - z^y_{k(n)}}{w^y_{k(n)}}\right)^2} \tag{31}$$

3.3 Reformalization of LGM-MCE Training Incorporating Sample-Dependent Weights

The definition of the geometric misclassification measure is directly influenced by the weighting of the samples in the misclassification measure space, and the LGM-MCE training procedure should be rewritten accordingly. Once we estimate the sample weights, we can in turn compute and use the weighted misclassification measures in the optimization process. We call the resulting procedure Weighted LGM-MCE training.

To define the weighted misclassification measures, we first review the general form of the Parzen window estimation in the misclassification measure space:

$$\frac{1}{h_y}\phi\Big(\frac{z - D_y(\boldsymbol{x}^y_n;\Lambda)}{h_y}\Big). \tag{32}$$

In the case of a variable width estimation, the window includes weight w^y_n assigned to sample $D_y(\mathbf{x}^y_n;\Lambda)$ in the misclassification measure space:

$$\frac{1}{h_y w^y_n}\phi\Big(\frac{z - D_y(\boldsymbol{x}^y_n;\Lambda)}{h_y w^y_n}\Big). \tag{33}$$

Following the same reasoning as in 2.3 leads to a redefinition of the smooth error count loss in the misclassification measure space:

$$\ell_y(D_y(\boldsymbol{x}^y_n;\Lambda)) = \int_0^\infty \frac{1}{h_y w^y_n}\phi\Big(\frac{z - D_y(\boldsymbol{x}^y_n;\Lambda)}{h_y w^y_n}\Big)dz. \tag{34}$$

For a Gaussian kernel, (34) can be identified as a sigmoid function of the geometric misclassification measure, which appears weighted by $\{w^y_n\}$ in this case:

$$\ell_y\left(\frac{D_y(\boldsymbol{x}^y_n;\Lambda)}{w^y_n}\right) = \frac{1}{1+\exp\left(-\alpha_y\frac{D_y(\boldsymbol{x}^y_n;\Lambda)}{w^y_n}\right)}. \tag{35}$$

Here we define weighted geometric misclassification measure $D^{(w)}_y(\boldsymbol{x}^y_n;\Lambda)$:

$$D^{(w)}_y(\boldsymbol{x}^y_n;\Lambda) \triangleq \frac{1}{w^y_n}D_y(\boldsymbol{x}^y_n;\Lambda), \text{where } w^y_n > 0. \tag{36}$$

In (36), $w^y_n > 1$ implies a weighted misclassification measure closer to zero than its original counterpart, in other words the sample is closer to the class boundary. This situation corresponds to a larger impact of the training sample on the LGM-MCE optimization process (Fig. 1), and also to a flatter Parzen width around the training sample in the misclassification measure space which implies more virtual samples surrounding this training sample.

The formulation of the variable Parzen estimation implies that higher weights are assigned to the samples located in areas where the density in the misclassification measure space is sparser, which likely corresponds to samples located near the class boundaries. To increase the reliability of the classification decision on future samples, the influence of such samples on the training should be weighted more heavily during the training, which is precisely the effect of the weighted geometric misclassification measure; conversely, $w^y_n < 1$ corresponds to a sample located in the higher density regions further from the class boundaries, where samples are less relevant to discriminate between the classes. In other words, the weighting of the misclassification measure can be seen as a way of increasing the robustness of LGM-MCE training at two levels:

1. The weighted widths during the risk estimate.
2. The weighted misclassification measures that directly influence the minimization of the empirical average loss (5).

The training procedure of Weighted LGM-MCE can be implemented in Algorithm 3.

4. Experimental Evaluation

4.1 Experimental Conditions

4.1.1 Classifier

To evaluate our proposed scheme, we adopted a multi-prototype classifier, where each class was modeled by multiple prototypes and its discriminant function was

```
Input: $\{\Lambda\}^{(0)}, e \leftarrow 0, E', E$
Output: $\{\Lambda\}^{(e+1)}$
1  while
   $(e < E) \| (convergence\ condition\ not\ reached)$ do
       /* Periodical loss smoothness
          estimation                                   */
2      if $e \equiv E' \pmod 0$ then
3          for $y \in [1, C])$ do
4              Initialize $\forall n \in [1, C_y], w_n^y \leftarrow 1$ and $h_y$ ;
               /* Equation 31 */
5              Estimate $\{w_n^y\}_{n=1}^{N_y}$ and $h_y$ ;
               /* Algorithms 2 and 1 */
6              $\alpha_y \leftarrow \frac{4}{\sqrt{2\pi} h_y}$ ;      /* Equation 11 */
7          end
8      end
       /* Probabilistic Descent optimization
          */
9      Shuffle $\Omega_N$ ;
10     for $n \in [1, N]$ do
11         $\Lambda^{(e+1)} \leftarrow \Lambda^{(e)} - \epsilon(t) \nabla_\Lambda l_y \left( D_y^w(\boldsymbol{x}_n^y; \Lambda) \right)$
12     end
13     $e \leftarrow e + 1$ ;
14 end
```

Algorithm 3: Weighted LGM-MCE training incorporating class-by-class automatic loss smoothness determination. Here, E is number of LGM-MCE training *epochs*, where epoch denotes the classifier parameter update sequence using all training samples once.

defined by the (minus) distance between an input and its *closest prototype*. Here, the closest prototype was one that had the minimum distance to input among the prototypes assigned to each class. Accordingly, the discriminant function for class C_j is given as

$$g_j(\boldsymbol{x}; \Lambda) = - \parallel \boldsymbol{x} - \boldsymbol{p}_j \parallel^2, \tag{37}$$

where $\boldsymbol{p}_j$ is C_j's closest prototype for $\boldsymbol{x}$. Then the corresponding geometric misclassification measure becomes:

$$D_y(\boldsymbol{x}; \Lambda) = \frac{\parallel \boldsymbol{x} - \boldsymbol{p}_y \parallel^2 - \parallel \boldsymbol{x} - \boldsymbol{p}_i \parallel^2}{\parallel \boldsymbol{p}_y - \boldsymbol{p}_i \parallel}, \tag{38}$$

where C_y, $\boldsymbol{p}_y$, C_i, and $\boldsymbol{p}_i$ are the correct class, its corresponding closest prototype, the best incorrect class, and its corresponding closest prototype for $\boldsymbol{x}$, respectively.

4.1.2 Dataset

We conducted all of our experiments on the Letter Recognition (LR) dataset, which consists of 16-dimensional feature vectors extracted from 20 000 character font-images of the English alphabet (26 classes). The dataset was divided into a training set (1000 samples) and a testing set (18 000 samples) for computing the classification rate of unknown data.

4.1.3 Optimization Method

We used the Probabilistic Descent method to minimize the average empirical loss of the LGM-MCE training. The learning parameter must still be set empirically. Because our analysis aims to analyze the robustness of the weighting scheme performed on the training set, we tuned this hyperparameter in regard to the training set instead of a traditional validation set. Practically, we swept range $\epsilon_0 = [2^{-5}, 2^5]$ by exponents of 2 and set the number of LGM-MCE epochs E to 10 000. For Weighted LGM-MCE, the update rule of prototype $\boldsymbol{p}_j$ based on training sample $\boldsymbol{x}_n^y$ becomes:

$$\boldsymbol{p}_j{}^{(t+1)} = \boldsymbol{p}_j{}^{(t)} - \epsilon_t \nabla_{\boldsymbol{p}_j} \ell_y \left(D_y^{(w)}(\boldsymbol{x}_n^y; \Lambda^{(t)}) \right) =$$
$$\boldsymbol{p}_j{}^{(t)} - \epsilon_t \ell'_y \left(\frac{1}{w_n^y} D_y(\boldsymbol{x}_n^y; \Lambda^{(t)}) \right) \frac{1}{w_n^y} \nabla_{\boldsymbol{p}_j} D_y(\boldsymbol{x}_n^y; \Lambda^{(t)}),$$

where $\epsilon_t = \epsilon_0(1 - \frac{t}{T})$, T denotes the total number of iteration steps over E epochs, and $\boldsymbol{x}_n^y$ is a training sample randomly extracted from Ω_N.

We repeated 10 weight updates and 100 CVML iterations for each set of weight values and tried ten different values from $E' = 10$ to $E' = 100$ at ten intervals.

4.2 Results and Considerations

4.2.1 Analysis of Variable Parzen Width Estimation

The observations in Fig. 2 for analyzing the convergence property of the sample-dependent Parzen width estimation are introduced in Algorithm 1. Indeed, even though the CVML procedure introduced in 3.2 guarantees the monotonic increase of each F_y for a fixed set of sample weights $\{w_n^y\}_{n=1}^{N_y}$, no specific guarantee is provided regarding the convergence of F_y from one sample weight update to another.

To analyze the algorithm fs convergence, we performed the observations displayed in Fig. 2.

1. Fig. 2(a) monitors likelihood F_0 relative to category 0 computed on the training data during the whole width estimation process (similar trends can be observed for the other 25 categories). In this experimental setting, four weight re-estimations performed every 100 iterations result in a total of $4 * 100 = 400$ algorithm iterations, the first 100 iterations of which correspond to the fixed-width Parzen estimation initialization, while each of the $3 * 100$ subsequent iterations correspond to EM optimization under fixed sample-dependent weight setting $\{w_k^0\}^{(l)}, \forall l \in [0, 3]$.

2. Fig. 2(c) focuses on one of the four successive EM optimizations (the same trend is strictly observed for all four EM repetitions). As expected from the EM optimization procedure, F_0 increases monotonically during EM optimization $[k * 100, (k + 1) * 100], \forall k \in [0, 3]$, which aims to improve the quality of the risk estimate based on finite training data, as described in (6).

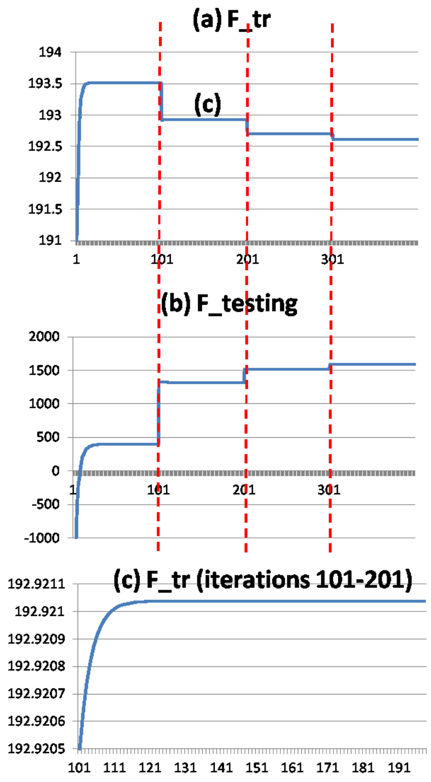

Figure 2: Evolution of likelihood F_0 (19) relative to category 0 during sample-dependent width estimation. Likelihood is observed at the end of each CVML estimation. Note that only four weight re-estimations are shown on the graph, which is sufficient to illustrate convergence of F_0: (a) Likelihood computed on training set; (b) Likelihood computed on testing set; (c) Detailed observation of the second CVML performed in (a).

3. Despite the maximization of F_0 inside each EM repetition, Fig. 2(a) showed a surprisingly neat decrease at every weight re-estimation. Furthermore, such decrease initially clearly overweighs the increase of F_0 during the previous EM optimization. A possible interpretation is that by assigning wider Parzen window widths to samples located in sparser density areas in the misclassification measure space, Algorithm 2 reduces the potential overfitting to the training distribution. To understand this, assume an extreme case where kernel width h_n^0 is set to 0 for each training sample z_n^0. This situation corresponds to a mere replication of the original finite training distribution for class 0. Hence, based on the density in the misclassification measure space, setting an appropriate degree of smoothness of existence around each original training sample can be seen as an attempt to increase the estimation quality over the whole (misclassification-measure-mapped) feature space. In other words, decreasing likelihood F_0 is a measure that improves the quality of the density estimation regarding unseen samples and the risk estimate.

4. To validate our interpretation, we monitored likelihood F_0 over the testing data during the weight optimization process performed on the training data, as shown in Fig. 2(b). As expected, each weight re-estimation (and the decrease of F_0 over the training data) corresponds to a significant increase of F_0 over unseen data, which validates the robustness of the sample-dependent weighting scheme.

4.2.2 Classification Performance of Weighted LGM-MCE

To validate the impact of the weighting scheme on the robustness of the training, we recorded the classification rates on the training and testing data in Fig. 3. The more detailed settings are provided in Table 1. The upper (resp. lower) panel shows the classification accuracy over training (resp. testing) data, for preceding LGM-MCE training using fixed-width loss smoothness estimation (red bars) and weighted LGM-MCE (Algorithm 3) (blue bars).

The results show that five prototypes are enough to achieve perfect accuracy on the training set, which likely corresponds to overfitting. Interestingly, Weighted LGM-MCE almost always achieves a higher score on testing data for $k > 3$, which can be interpreted as reducing the overfitting incurred by a higher number of prototypes.

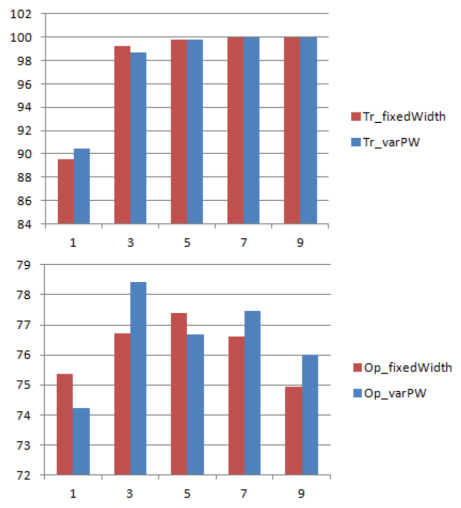

Figure 3: Comparison of classification rates on LR data between fixed-width estimation LGM-MCE and Weighted (variable width) LGM-MCE).

Though more systematic experiments exploring different settings are necessary, the results in Table 1 show a tendency for the Weighted LGM-MCE to outperform its standard counterpart in terms of generalization performance to unseen samples, which validates the robustness of the sample-dependent weighting scheme.

Table 1: Details of classification rates on LR data between fixed-width estimation LGM-MCE and Weighted (sample-dependent width) LGM-MCE. For all prototype settings, value of learning parameter μ_0 leading to best performance on the training set is provided.

Classification rates on training data

k	Fixed-width (ϵ_0)	Variable width (ϵ_0)
1	89.50 (2^{-4})	90.50 (2^{-5})
3	99.20 (2^{-3})	98.70 (2^{-1})
5	99.80 (2^{-1})	99.80 (2^{-2})
7	100.0 (2^{-1})	100.0 (2^{-1})
9	100.0 (2^{-1})	100.0 (2^{-1})

Classification rates on testing data

k	Fixed-width (ϵ_0)	Variable width (ϵ_0)
1	75.36	74.24
3	76.73	78.44
5	77.38	76.70
7	76.62	77.46
9	74.93	75.99

5. Conclusion

We proposed a new LGM-MCE scheme that introduced a density-based weighting of samples in geometric misclassification measure space. By automatically assigning an appropriate degree of smoothness to each training sample in the geometric misclassification measure space, not only does the objective function (i.e., empirical average loss) become closer to the ideal goal of classifier (i.e., Bayes risk) but the optimization process itself is also refined by incorporating the relevance of each training sample while updating the classifier parameters. Experimental results confirmed the increased quality of the Bayes risk estimate using a weighting scheme as well as the possible increase of accuracy over unseen samples of the Weighted LGM-MCE-trained classifier.

6. Acknowlegment

This work was supported in part by JSPS Grants-in-Aid for Scientific Research No. 26280063 and MEXT-Supported Program "Driver-in-the-Loop".

References

[1] H. Watanabe, S. Katagiri, M. Ohsaki, K. Yamada, E. Dermott, A. Nakamura, S. Watanabe, and M. Ohsaki, "Minimum classification error training using geometric-margin-based misclassification measure," *IEICE Trans. D*, vol. J94-D, no. 10, pp. 1664-1675, 2010 (in Japanese).

[2] T. Ohashi, J. Tokuno, H. Watanabe, S. Katagiri, and M. Ohsaki, "Automatic loss smoothness determination for large geometric margin minimum classification error training," *Proc. TENCON 2011*, pp. 260-264, 2010.

[3] K. Ota, S. Katagiri, H. Watanabe, and M. Ohsaki, "Minimum classification error training to automatically determine loss smoothness common to all classes," *Master thesis (in Japanese)*, 2014.

[4] B.W. Silvermann, "Density estimation for statistics and data analysis," *Chapman and Hall/CRC*, 1986.

[5] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery (2)*, pp. 121-167, 1998.

[6] S. Katagiri, B-W Huang, C-H Lee, "Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method," *Proc. of the IEEE*, vol. 86, pp. 2345-2373, November 1998.

[7] C. Igel, M. Hasken, "Improving the Rprop algorithm," *Proc. of the Second International Symposium on Neural Computation*, pp. 115-121, 2000.

[8] H. Watanabe, J. Tokuno, T. Ohashi, S. Katagiri, M. Ohsaki, S. Matsuda, and H. Kashioka, "Minimum classification error training incorporating automatic loss smoothness determination," *Journal of Signal Processing Systems*, Vol.74, No.3, pp.311-322, Springer, 2014.

[9] R. Duda, P. Hart, and G. Stork, "Nonparametric techniques". In R. Duda, P. Hart, and G. Stork (2nd ed.) Pattern Classification, pp. 164-172, 2001.

[10] G. Terrel, and D. Scott, "Variable kernel density estimation," *Annals of Statistics*, 20(3), pp 1236-1265, 1992.

[11] A.W. Bowman, "An alternative method of cross-validation for the smoothing of density estimates," *Oxford Journals*, 1984.