Batch Map Extensions of the Kernel-based Maximum Entropy Learning Rule

Temujin Gautama, and Marc M. Van Hulle, Senior member, IEEE

Abstract

In this letter, two batch map extensions are described for the kernel-based maximum entropy learning rule. In the first, the weights are iteratively set to weighted component-wise medians, while in the second the generalised median is used, enabling kMER to process symbolic data. Simulations are performed to illustrate the extensions.

Index Terms

Batch map, kMER, generalised median

I. INTRODUCTION

Batch Map extensions of Kohonen’s Self-Organising Map (SOM; [1]) provide a considerable speed-up of the SOM training procedure, by replacing the incremental weight updates by an iterative scheme that sets the weight vector of each neuron to a weighted mean of the training data (weighted according to Voronoi membership and neighbourhood function). The Batch Map SOM (BM-SOM) can be interpreted as an extension of the Linde-Buzo-Gray algorithm for vector quantisation [2], and can be related to an expectation maximisation (EM) algorithm [3]. The BM-SOM has been further modified using the generalised median, allowing the algorithm to process symbolic data when a matrix of dissimilarities between data points can be computed [4]. In this letter, we propose two extensions of the kernel-based maximum entropy learning rule (kMER; [5]), namely the batch map versions using the component-wise weighted median, and using the generalised weighted median.

Both authors are with the Laboratorium voor Neuro- en Psychofysiologie, K.U.Leuven, Leuven, Belgium. This work has been supported by research grants received from the Belgian Fund for Scientific Research – Flanders (G.0248.03 and G.0234.04), the Flemish Regional Ministry of Education (Belgium) (GOA 2000/11), and the European Commission, 5th framework programme (QLG3-CT-2000-30161 and IST-2001-32114) and 6th framework programme (IST-001917).
II. METHODS

A. Incremental kMER

Contrary to Kohonen’s SOM, the kernel-based maximum entropy rule (kMER) allows the activation of multiple neurons for a given input data point. In the original kMER algorithm, neuron $i$, with weight vector $w_i$, is ‘activated’ by data point $v_p$ when $\|w_i - v_p\| < \sigma_i$, where $\sigma_i$ is the radius of neuron $i$, which defines a hyperspherical activation region, $S_i$. The membership function, $M_i(v_p)$, is defined as unity when neuron $i$ is activated by data point $v_p$, and zero otherwise. When there are no neurons active for a given input, the neuron that is positioned closest to that data point (the winner in the SOM terminology) is defined active. Furthermore, a fuzzy membership function is defined as $\Xi_i(v_p) = \sum_{i=1}^{N} M_i(v_p)$. The incremental learning schemes for the weights and radii of neuron $i$ are the following:

$$\Delta w_i = \eta \sum_{j=1}^{N} \Lambda(i,j,\sigma_\lambda) \Xi_i(v_p) \text{sign}(v_p - w_i)$$

$$\Delta \sigma_i = \eta \left( \rho \sum_{i=1}^{N} M_i(v_p) - M_i(v_p) \right),$$

where $\eta$ is the learning rate factor, $N$ is the number of neurons, $\sigma_\lambda$ is the neighbourhood range of the neighbourhood function $\Lambda(\cdot)$, $\rho = \frac{\rho N}{N-\rho}$ is a parameter controlling the overlap between activation regions of the neurons, and the sign function is taken component-wise. The neighbourhood function is usually defined as a unit-height Gaussian function with standard deviation $\sigma_\lambda$ (in lattice coordinates), and the cooling scheme, which defines the speed at which the neighbourhood influence between neurons becomes smaller during training, is defined by the evolution of $\sigma_\lambda$ during training, e.g., exponentially decreasing:

$$\sigma_\lambda(t) = \sigma_{\lambda,0} \exp \left( -2 \gamma \frac{t}{t_{\max}} \right),$$

with $\gamma$ controlling the cooling rate (small values of $\gamma$ yield slow cooling schemes; a standard value is $\gamma = 1$ [5]). We refer to Ref. [6] for an iterative method to find a sufficiently slow cooling scheme.

B. Batch Map kMER

It can be shown that when the neighbourhood function vanishes, a weight vector converges to the median of the input samples $v_p$ that activate the corresponding neuron (median taken per dimension), weighted by the fuzzy membership function $\Xi_i(v_p)$ [5]. Indeed, for that configuration of the weights, the sum of the weight updates (in the absence of the neighbourhood function $\Lambda(i,j,\sigma_\lambda)$ and summing over the data points in $S_i$) is zero. Extending this result to incorporate the influence of the neighbourhood
function, the sum of the weight updates for neuron \(i\) is zero when \(w_i\) is equal to the weighted median\(^1\) of all data points, weighted by \(\left( \sum_{j=1}^{N} \Lambda(i, j, \sigma_{\lambda}) \Xi_i(v_p) \right) (p = 1, \ldots, M)\). The radius of each neuron converges to the value for which there are \(\rho M / N\) data points in its activation region. Similar to the BM-SOM, the Batch Map kMER update rule (BM-kMER) can be defined as follows for each neuron \(i\):

1) the new weight vector \(w_i\) is the weighted median (per dimension) of all data points \(v_p\), weighted by \(\left( \sum_{j=1}^{N} \Lambda(i, j, \sigma_{\lambda}) \Xi_i(v_p) \right)\) (note that in the absence of activation for a given data point, the nearest neuron is made active);

2) the new radius \(\sigma_i\) is set equal to the distance between \(w_i\) and the \(((\rho M / N)\)-th nearest data point.

These two steps are iterated while the neighbourhood range \(\sigma_{\lambda}\) is decreased at every iteration (following a given cooling scheme). In this way, the learning rate factor, \(\eta\), is no longer required, and training is only influenced by the cooling rate \((\gamma)\), the length of training \((t_{\text{max}})\) and the degree of activation overlap \((\rho)\).

To illustrate the BM-kMER, a synthetic data set is generated consisting of \(M = 1000\) two-dimensional data points with two clusters (unit-variance Gaussians centred at (3,0) and (-2,1)), and a \([10 \times 10]\) lattice is trained using the BM-kMER with \(\rho = 2\). Rather than starting from an ordered initial lattice, as suggested in [1], an initial training of 50 iterations with maximal neighbourhood range, \(\sigma_{\lambda,0} = 5\), which is half of the lattice size along one dimension, is performed, starting from a random lattice (weights drawn from a standard normal distribution and radii set to 0.01). Note that this ‘initialisation’ approach is also the first step in the monitoring process proposed in [6]. The result is an unfolded map in the centre of the distribution, as shown in Fig. 1A. The training session is resumed for \(t_{\text{max}} = 100\) iterations with a cooling rate of \(\gamma = 0.5\) (Eq. 2; note that \(t = 0\) refers to the iteration after the initial training). The lattices during training are shown in Figs. 1B–F for \(t = 10, 20, 40, 47, 70\), respectively. The overlap variability metric (OV; [6]) is used for determining the point at which to terminate training (leaving a residual neighbourhood influence and map smoothness), yielding the configuration of Fig. 1E.

The speed of convergence of the BM-kMER is compared to that of the standard incremental scheme by plotting the evolution of the weights during the initial training. Both the incremental and the batch map version are initialised with the same random configuration, and the learning rate for the first is set to \(\eta = 0.004\) (this value was determined in a trial-and-error manner as the largest value for which

---

\(^1\)The weighted median of values \(\{x_k\}\) with corresponding weights \(\{W_k\}\) is obtained by sorting the values \(\{x_k\}\), and observing the cumulative sum of the weights corresponding to the sorted series. The weighted median is the value with the smallest index for which this cumulative sum exceeds \(\sum W_k / 2\).
the lattice still showed smoothly convergent behaviour, \textit{i.e.}, without oscillations). It can be observed from Fig. 2A and 2B that the BM-kMER converges much faster (the figures show the traces of the first component of the weight vectors of each neuron). This can be quantitatively assessed by determining the point where all scalar weight coefficients are close (absolute difference smaller than 0.05) to their asymptotic value (estimated as the average over the final 10 iterations), \textit{i.e.}, 16 for the BM-kMER and 44 for the incremental kMER. For some weights, there is still a slight variability for the BM-kMER (Fig. 2B), which is due to the interplay between the fuzzy membership influence (due to the radii) and the neuron positions, and the discrete nature of the median. Indeed, a data point is either inside or outside the activation region of a given neuron. Therefore, when it is at the border of the activation region, it can lead to a metastable configuration, similar to that often observed in the well-known \(k\)-means algorithm (where a data point is either inside or outside a Voronoi region).

\textbf{C. Batch Map kMER – Generalised Median}

One of the attractive aspects of Batch Map type algorithms, is their extension to process non-metric data \cite{1}. In the SOM case, at every iteration, each weight vector, \(w_i\), is replaced by the generalised median of the data points assigned to the neurons in the topological neighbourhood of neuron \(i\), optionally weighted by the neighbourhood function \cite{4}, instead of the weighted mean in the standard BM-SOM. In this way,
Fig. 2. Evolution of the first component of the weight vectors over time for the incremental kMER (A), the BM-kMER (B) and the BM-kMER with generalised median (C).

all neuron positions coincide with actual data points, and the map can be trained by means of only the $[M \times M]$ array of pairwise distances (dissimilarity matrix). Furthermore, this matrix is not necessarily derived from a metric data space, but can contain any measure of dissimilarity between data points.

In the present kMER case, the weighted median per dimension is replaced by the (weighted) generalised median, i.e., the data point $v^*$ for which the weighted distance to the other data points is smallest:

$$i^* = \text{argmax}_i \sum_{p=1}^{M} F_p \text{dist}(i, p),$$

with $F_p$ the weighting factor for data point $v_p$. Thus, the update rules for non-metric data are:

1) the new weight vector $w_i$ is determined as the weighted generalised median of all data points $v_p$, weighted by $\left( \sum_{j=1}^{N} \Lambda(i, j, \sigma_{\lambda}) \Xi_i(v_p) \right)$ (in the absence of activation for a given data point, the nearest neuron is made active);

2) the new radius $\sigma_i$ is set equal to the distance between $w_i$ and the $\left( \frac{M}{N} \right)$-th nearest data point.

The method is applied to the previous two-dimensional data set. Albeit these data are positioned in a metric space, the method only uses the $[1000 \times 1000]$ array of pairwise Euclidean distances between data points, and not the actual positions in data space. Illustrating the method on this data set, rather than on a fully non-metric one offers the advantage that visualisation of the lattice is still possible in data space.

The initial training (the weights are initialised to the first 100 data points of a random permutation of the data set) is again performed during 50 iterations with a fixed neighbourhood range ($\sigma_{\lambda,0} = 5$), resulting in the configuration shown in Fig. 3A (note that all neuron positions correspond to a data point). The training session is resumed for $t_{\text{max}} = 100$ iterations with a cooling rate of $\gamma = 0.5$ (Eq. 2). The evolution of the lattice during training (at $t = 10, 20, 40, 54, 70$) is shown in Fig. 3B–F, and the configuration with minimal overlap variability is that of Fig. 3E.

The traces of the neuron positions during the initial training are shown in Fig. 2C (for all neurons, the traces of the first component of the weight vectors are shown). Similar to the BM-kMER case, the
Fig. 3. Lattice configuration after initial training (A) and during BM-kMER training with the generalised median at $t = 10, 20, 40, 54, 70$, respectively in panels B–F. The configuration in panel E corresponds to the minimal overlap variability value (see Section II-C).

weights show fast convergence, but six neurons show an oscillatory behaviour for $t > 15$. We have observed in other simulations that this oscillatory behaviour is no longer present for low neighbourhood values such as those at the minimal OV (results not shown). It is, again, due to the interplay of the fuzzy membership influence and the discrete nature of the generalised median (even to a greater extent than the BM-kMER case with the component-wise median). The point where the weight coefficients reach their asymptotical value is after 14 iterations (not taking into account the oscillating coefficients).

III. Real-World Example

As an example of the proposed Batch Map kMER, we consider the “digit” data set described in [7], which consists of features derived from handwritten numerals (“0”–“9”) with 200 data points for each class of numerals. The 649-dimensional data are standardised per dimension and are used to train a $[7 \times 7]$ lattice. Training parameters are the following: $t_{\text{max}} = 200$, $\gamma = 1$, $\rho = 2$. The initial weights are taken randomly from a standard normal distribution and the radii are all set to 0.01 (both for the kMER and the BM-kMER). Initial training with the BM-kMER is performed during 20 iterations with a fixed neighbourhood range of $\sigma_{\lambda,0} = 3.5$, after which the cooling scheme of Eq. 2 is used. During this training session, the OV is evaluated after each iteration and the training session is terminated when the OV stops changing (when the standard deviation over the previous ten iterations is smaller than 0.01; this heuristic was employed rather than selecting the minimum of the OV-curve, as no clear minimum was observed).
Since the “digit” data set consists of labelled data points, the following approach has been adopted to illustrate the quality of the obtained map. For each neuron \( i \), the labels of the data points within its activation region \( S_i \) are determined, and the most frequently occurring label \( L_i \) is used for grey-level coding the lattice in Fig. 4A (each grey level corresponds to one of the ten labels). It is likely that the map has preserved the topology in mapping the high-dimensional data to the two-dimensional lattice, since the 10 classes comprise contiguous regions. Furthermore, the ratio of data points with label \( L_i \) within the activation region of neuron \( i \), are plotted as an indication of the ‘confidence’ with which the neuron represent the class (Fig. 4B; white corresponds to unity, black to zero). Only a single neuron has a confidence score of 0.51, and the others have confidences exceeding 0.75. For comparison, the same training (\( t_{\text{max}} = 200, \gamma = 1, \rho = 2 \)) has been performed with the incremental kMER scheme, with a learning rate of \( \eta = 1/M = 0.0005 \), but the length of the initial training is doubled. The obtained labelled lattice (Fig. 4C) also shows an unfolded lattice, with confidences slightly lower than the BM-kMER case, but similar (lowest value of 0.42 and 13 neurons with confidences lower than 0.75). Therefore, we tentatively conclude that the representation of the BM-kMER lattice is slightly better than that of the incremental kMER.

The advantage of the BM-kMER becomes clear when the standardisation of the data is absent. Note that standardisation is performed for each dimension separately, due to which the effect can be interpreted as a linear stretching of certain dimensions with respect to other dimensions. Since the kMER algorithm uses circular symmetrical activation regions, and, in the case of the incremental version, weight updates of fixed and identical size in each dimension, this heavily influences the learning behaviour, and in this case, makes the training considerably more difficult due to the wide range of values in the data set: the minimal (maximal) standard deviation over dimensions is 0.038 (3757.634). The training parameters for the BM-kMER were \( t_{\text{max}} = 1000, \gamma = 0.5 \) and \( \rho = 2 \), and an initial training phase of 20 iterations. The obtained lattice is shown in Fig. 4D, and it can be observed that the lattice consists of contiguously labelled regions, indicating that the lattice is unfolded and that the mapping retains the topography. The training of the incremental kMER on the non-standardised data is very difficult, since it is heavily dependent on the learning rate. Due to the component-wise sign update and the large range of different variances per dimension, it would probably be necessary to use separate learning rates per dimension, but this has not been further pursued. With a fixed learning rate, it is nearly impossible to obtain a smoothly convergent behaviour (without too large oscillations) at an acceptable speed. As an illustration of the convergence speed (with a learning rate of 0.01, which is at the limit of stability), the length of the initial training phase needs to be increased to 200 iterations (a 10-fold increase with respect to the BM-kMER.
IV. Conclusions

In this letter, we have proposed two extensions of the kernel-based maximum entropy learning rule (kMER; [5]). The first is the Batch Map kMER (BM-kMER), which, at every iteration, sets the weights to a component-wise weighted median of the input data (weighted by the fuzzy membership and the neighbourhood function). The second extension uses the generalised, rather than the component-wise median, thus making it possible to process symbolic data when a matrix of dissimilarity measures between data points can be computed. In both cases, there is no need to define a learning rate parameter. The speed of convergence of both extensions is considerably higher than that of the incremental scheme, as expected, which becomes increasingly important for data sets with large differences in variance across dimensions.

As an illustration, the BM-kMER is compared to the incremental kMER for a high-dimensional real-world example, namely a data set consisting of 649-dimensional feature vectors derived from handwritten digits [7]. In a qualitative manner, it has been shown that, when the data are standardised per dimension, the lattice obtained using the BM-kMER constitutes a slightly better representation of the data set than that obtained with the incremental kMER. It is likely that a more finely tuned training session, such as that derived from monitoring algorithms [6], would yield an equally good representation. However, when the data were not standardised, the differences in variance across dimensions were found to have a large influence on the training of the incremental kMER, making it impractically slow. Since the BM-kMER is faster than the incremental scheme, does not require a learning rate, and suffers less of large variance case). The remainder of the training is intractably slow (simulations were performed on a Pentium 4 processor running Matlab 6.5 in a Linux environment). Indeed, if a neuron component has to traverse one standard deviation along the dimension with the highest variance using a learning rate of $\eta = 0.01$, this would take 375,763 iterations. Therefore, the kMER training on non-standardised data has not been performed.
differences between dimensions, the combination with a monitoring scheme is expected to yield a fast and robust tool for performing an exploratory data analysis.

REFERENCES


