



Prototypes Within Minimum Enclosing Balls

Christian Bauckhage^{1,2,3} , Rafet Sifa^{1,2}, and Tiansi Dong³

¹ Fraunhofer Center for Machine Learning, Sankt Augustin, Germany

² Fraunhofer IAIS, Sankt Augustin, Germany

{christian.bauckhage,rafet.sifa,tiansi.dong}@iais.fraunhofer.de

³ B-IT, University of Bonn, Bonn, Germany

Abstract. We revisit the kernel minimum enclosing ball problem and show that it can be solved using simple recurrent neural networks. Once solved, the interior of a ball can be characterized in terms of a function of a set of support vectors and local minima of this function can be thought of as prototypes of the data at hand. For Gaussian kernels, these minima can be naturally found via a mean shift procedure and thus via another recurrent neurocomputing process. Practical results demonstrate that prototypes found this way are descriptive, meaningful, and interpretable.

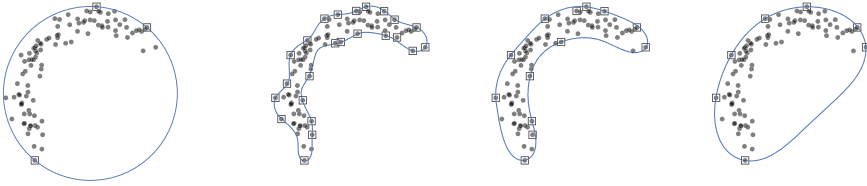
1 Introduction

The problem of characterizing data in terms of prototypes commonly arises in contexts such as clustering, latent component analysis, manifold identification, or classifier training [15, 16, 19, 20, 23]. Ideally, prototype identification should be computationally efficient and yield representative and interpretable results either for meaningful downstream processing or for assisting analysts in their decision making. In this paper, we discuss a two-stage approach based on kernel minimum enclosing balls and their characteristic functions that meets all these requirements.

Minimum enclosing balls (MEBs) are central to venerable techniques such as support vector clustering [3], or support vector data description [17]. More recently, balls have shown remarkable success in structuring deep representation learning [6, 7, 14]. Here, we revisit the kernel MEB problem and discuss how to solve it using simple recurrent neural networks. Resorting to recent work [1] which showed that recurrent neural networks can accomplish Frank-Wolfe optimization [9], we show how the Frank-Wolfe algorithm allows for finding MEBs and how this approach can be interpreted in terms of reservoir computing.

The solution to the kernel MEB problem consists in a set of support vectors that define the surface of a ball in a high dimensional feature space. Its interior, too, can be characterized in terms of a function of its support vectors. Local minima of this function coincide with representative and easily interpretable prototypes of the given data and we show that, for balls computed using Gaussian kernels, these minima are naturally found via generalized mean shifts [4, 10].

Since the mean shift procedure, too, can be interpreted in terms of reservoir computing, the approach we present in this paper constitutes an entirely neuro-computing based method for prototype extraction.



(a) Euclidean MEB (b) Gaussian kernel MEBs for growing scale parameters

Fig. 1. A 2D data set, its Euclidean MEB, and several Gaussian kernel MEBs. Squares indicate which data points support the surface of the corresponding ball

2 Minimum Enclosing Balls in Data- and Feature Space

In order for our presentation to be self-contained, we begin with a brief review of the minimum enclosing ball (MEB) problem.

Given a data matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$, the minimum enclosing ball problem asks for the smallest Euclidean m -ball $\mathcal{B}(\mathbf{c}, r)$ with center $\mathbf{c} \in \mathbb{R}^m$ and radius $r \in \mathbb{R}$ that contains each of the given data points \mathbf{x}_i .

Understood as an inequality constrained convex minimization problem, the *primal MEB problem* is to solve

$$\begin{aligned} \mathbf{c}_*, r_* &= \underset{\mathbf{c}, r}{\operatorname{argmin}} && r^2 \\ \text{s. t.} &&& \|\mathbf{x}_i - \mathbf{c}\|^2 - r^2 \leq 0 \quad i = 1, \dots, n. \end{aligned} \tag{1}$$

Evaluating the Lagrangian and Karush-Kuhn-Tucker conditions for (1) yields the corresponding *dual MEB problem*

$$\begin{aligned} \boldsymbol{\mu}_* &= \underset{\boldsymbol{\mu}}{\operatorname{argmax}} && \boldsymbol{\mu}^\top \mathbf{z} - \boldsymbol{\mu}^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\mu} \\ \text{s. t.} &&& \boldsymbol{\mu}^\top \mathbf{1} = 1 \\ &&& \boldsymbol{\mu} \succeq \mathbf{0} \end{aligned} \tag{2}$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$ is a vector of Lagrange multipliers, $\mathbf{0}, \mathbf{1} \in \mathbb{R}^n$ denote vectors of all zeros and ones, and the entries of $\mathbf{z} \in \mathbb{R}^n$ are given by $z_i = \mathbf{x}_i^\top \mathbf{x}_i$.

The Karush-Kuhn-Tucker conditions further reveal that, once (2) has been solved, center and radius of the sought after ball amount to

$$\mathbf{c}_* = \mathbf{X} \boldsymbol{\mu}_* \tag{3}$$

$$r_* = \sqrt{\boldsymbol{\mu}_*^\top \mathbf{z} - \boldsymbol{\mu}_*^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\mu}_*}. \tag{4}$$

Note that the given data points enter the problem in (2) only in form of inner products with other data points because $\mathbf{X}^\top \mathbf{X}$ is an $n \times n$ Gramian with entries $(\mathbf{X}^\top \mathbf{X})_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$ and $\mathbf{z} = \text{diag}[\mathbf{X}^\top \mathbf{X}]$. The dual thus allows for invoking the kernel trick where inner products are replaced by non-linear kernel functions so as to implicitly solve the problem in a high dimensional feature space.

Hence, letting $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ be a Mercer kernel, we introduce $\mathbf{K} \in \mathbb{R}^{n \times n}$ where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{k} \in \mathbb{R}^n$ such that $\mathbf{k} = \text{diag}[\mathbf{K}]$ and obtain the *kernel MEB problem*

$$\begin{aligned} \boldsymbol{\mu}_* &= \underset{\boldsymbol{\mu}}{\text{argmax}} \quad \boldsymbol{\mu}^\top \mathbf{k} - \boldsymbol{\mu}^\top \mathbf{K} \boldsymbol{\mu} \\ \text{s. t.} \quad &\boldsymbol{\mu}^\top \mathbf{1} = 1 \\ &\boldsymbol{\mu} \succeq \mathbf{0}. \end{aligned} \tag{5}$$

Once (5) has been solved, the radius of the minimum enclosing ball in feature space can be computed analogously to (4), namely

$$r_* = \sqrt{\boldsymbol{\mu}_*^\top \mathbf{k} - \boldsymbol{\mu}_*^\top \mathbf{K} \boldsymbol{\mu}_*}. \tag{6}$$

However, the center of the feature space ball cannot be computed similarly since (3) does not lend itself to the kernel trick. Nevertheless, computing

$$\mathbf{c}_*^\top \mathbf{c}_* = \boldsymbol{\mu}_*^\top \mathbf{K} \boldsymbol{\mu}_*. \tag{7}$$

still allows for checking whether or not an arbitrary $\mathbf{x} \in \mathbb{R}^m$ resides within the kernel MEB of the given data. This is because the inequality $\|\mathbf{x} - \mathbf{c}_*\|^2 \leq r_*^2$ can be rewritten as

$$K(\mathbf{x}, \mathbf{x}) - 2 \boldsymbol{\kappa}^\top \boldsymbol{\mu}_* + \boldsymbol{\mu}_*^\top \mathbf{K} \boldsymbol{\mu}_* \leq \boldsymbol{\mu}_*^\top \mathbf{k} - \boldsymbol{\mu}_*^\top \mathbf{K} \boldsymbol{\mu}_* \tag{8}$$

where $\boldsymbol{\kappa} \in \mathbb{R}^n$ in the second term on the left has entries $\kappa_i = K(\mathbf{x}, \mathbf{x}_i)$.

Figure 1 compares the Euclidean minimum enclosing ball of a set of 2D data to kernel minimum enclosing balls computed using Gaussian kernels

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\lambda^2}\right) \tag{9}$$

with different scale parameters λ . In order to visualize the surfaces of the feature space balls in the original data space, we considered the function

$$f(\mathbf{x}) = \sqrt{K(\mathbf{x}, \mathbf{x}) - 2 \boldsymbol{\kappa}^\top \boldsymbol{\mu}_* + \boldsymbol{\mu}_*^\top \mathbf{K} \boldsymbol{\mu}_*} - \sqrt{\boldsymbol{\mu}_*^\top \mathbf{k} - \boldsymbol{\mu}_*^\top \mathbf{K} \boldsymbol{\mu}_*} \tag{10}$$

and highlighted the contour for which $f(\mathbf{x}) = 0$. Note that $f(\mathbf{x})$ can be seen as a characteristic function of the corresponding MEB \mathcal{B} , because $f(\mathbf{x}) \leq 0 \Leftrightarrow \mathbf{x} \in \mathcal{B}$ and $f(\mathbf{x}) > 0 \Leftrightarrow \mathbf{x} \notin \mathcal{B}$.

Finally, we note that those data points \mathbf{x}_i which support the surface of an MEB \mathcal{B} in data- or in feature space are easily identified. This is because only if \mathbf{x}_i resides on the surface of the ball will its Lagrange multiplier μ_{i*} exceed zero; for points inside the ball the inequality constraints in (2) or (5) are inactive and their multipliers vanish. Below, we will refer to points whose multipliers exceed zero as the *support vectors* \mathbf{s}_j of \mathcal{B} .

Algorithm 1. Frank-Wolfe algorithm for (11)

initialize a feasible point in Δ^{n-1} , for instance

$$\boldsymbol{\mu}_0 = \frac{1}{n} \mathbf{1}$$

for $t = 0, \dots, t_{\max}$ **do**

determine the step direction

$$\boldsymbol{\nu}_t = \underset{\boldsymbol{\nu} \in \Delta^{n-1}}{\operatorname{argmin}} -\boldsymbol{\nu}^\top \nabla \mathcal{D}(\boldsymbol{\mu}_t)$$

update the current estimate

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \frac{2}{t+2} [\boldsymbol{\nu}_t - \boldsymbol{\mu}_t]$$

3 Neural Computation of Minimum Enclosing Balls

Next, we discuss how the Frank-Wolfe algorithm [9] solves the kernel MEB problem and how this approach can be interpreted in terms of reservoir computing.

Observe that the kernelized dual Lagrangian $\mathcal{D}(\boldsymbol{\mu}) = \boldsymbol{\mu}^\top \mathbf{k} - \boldsymbol{\mu}^\top \mathbf{K} \boldsymbol{\mu}$ in (5) is concave so that $-\mathcal{D}(\boldsymbol{\mu}) = \boldsymbol{\mu}^\top \mathbf{K} \boldsymbol{\mu} - \boldsymbol{\mu}^\top \mathbf{k}$ is convex. We may therefore rewrite the maximization problem in (5) in terms of a minimization problem

$$\boldsymbol{\mu}_* = \underset{\boldsymbol{\mu} \in \Delta^{n-1}}{\operatorname{argmin}} \boldsymbol{\mu}^\top \mathbf{K} \boldsymbol{\mu} - \boldsymbol{\mu}^\top \mathbf{k} \quad (11)$$

where we also exploited that the non-negativity and sum-to-one constraints in (5) require any feasible solution to reside in the standard simplex $\Delta^{n-1} \subset \mathbb{R}^n$.

Written as in (11), our problem is clearly recognizable as an instance of a convex minimization problem over a compact convex set and we note that the Frank-Wolfe algorithm provides a simple iterative solver for this setting.

Algorithm 1 shows how it specializes to our context: Given an initial guess $\boldsymbol{\mu}_0$ for the solution, each iteration of the algorithm determines which $\boldsymbol{\nu}_t \in \Delta^{n-1}$ minimizes the inner product $\boldsymbol{\nu}^\top \nabla \mathcal{D}(\boldsymbol{\mu}_t)$ and applies a conditional gradient update $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \eta_t (\boldsymbol{\nu}_t - \boldsymbol{\mu}_t)$ where the step size $\eta_t = \frac{2}{t+2} \in [0, 1]$ decreases over time. This way, updates will never leave the feasible set and the efficiency of the algorithm stems from the fact that it turns a quadratic problem into a series of simple linear problems.

Next, we build on recent work [1] and show how Frank-Wolfe optimization for the kernel MEB problem can be implemented by means of rather simple recurrent neural networks.

For the gradient of the negated dual Lagrangian $-\mathcal{D}(\boldsymbol{\mu})$, we simply have $-\nabla \mathcal{D}(\boldsymbol{\mu}) = 2 \mathbf{K} \boldsymbol{\mu} - \mathbf{k}$ so that each iteration of the Frank-Wolfe algorithm has to compute

$$\boldsymbol{\nu}_t = \underset{\boldsymbol{\nu} \in \Delta^{n-1}}{\operatorname{argmin}} \boldsymbol{\nu}^\top [2 \mathbf{K} \boldsymbol{\mu}_t - \mathbf{k}]. \quad (12)$$

The objective function in (12) is linear in $\boldsymbol{\nu}$ and needs to be minimized over a compact convex set. Since minima of a linear functions over compact convex sets are necessarily attained at a vertex, the solution of (12) must coincide with a vertex of Δ^{n-1} . Since the vertices of the standard simplex in \mathbb{R}^n correspond to the standard basis vectors $\mathbf{e}_j \in \mathbb{R}^n$, we can cast (12) as

$$\boldsymbol{\nu}_t = \underset{\mathbf{e}_j \in \mathbb{R}^n}{\operatorname{argmin}} \mathbf{e}_j^\top [2\mathbf{K}\boldsymbol{\mu}_t - \mathbf{k}] \approx \mathbf{g}_\beta(2\mathbf{K}\boldsymbol{\mu}_t - \mathbf{k}). \quad (13)$$

where $\mathbf{g}_\beta(\mathbf{x})$ introduced in the approximation on the right of (13) represents the vector-valued softmin operator. Its i -th component is given by

$$(\mathbf{g}_\beta(\mathbf{x}))_i = \frac{e^{-\beta x_i}}{\sum_j e^{-\beta x_j}} \quad (14)$$

and we note that

$$\lim_{\beta \rightarrow \infty} \mathbf{g}_\beta(\mathbf{x}) = \underset{\mathbf{e}_j \in \mathbb{R}^n}{\operatorname{argmin}} \mathbf{e}_j^\top \mathbf{x} = \mathbf{e}_i. \quad (15)$$

Based on the relaxed optimization step in (13), we can therefore rewrite the Frank-Wolfe updates for our problem as

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \eta_t [\boldsymbol{\nu}_t - \boldsymbol{\mu}_t] \quad (16)$$

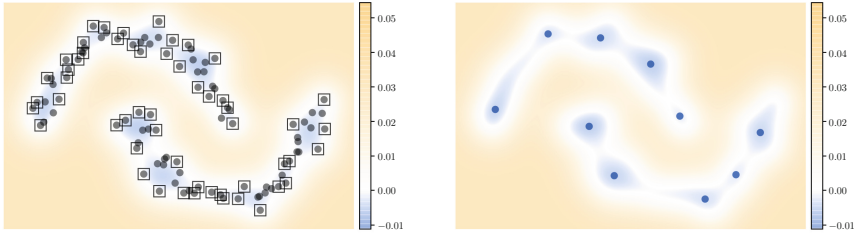
$$= (1 - \eta_t) \boldsymbol{\mu}_t + \eta_t \boldsymbol{\nu}_t \quad (17)$$

$$\approx (1 - \eta_t) \boldsymbol{\mu}_t + \eta_t \mathbf{g}_\beta(2\mathbf{K}\boldsymbol{\mu}_t - \mathbf{k}). \quad (18)$$

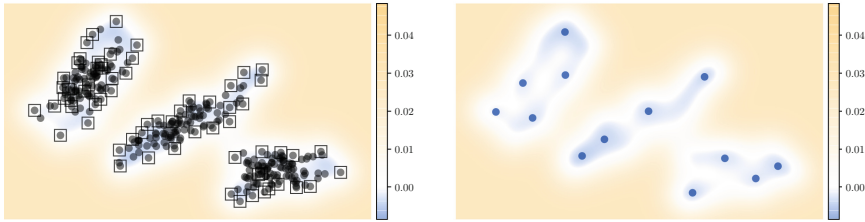
Choosing an appropriate parameter β for the softmin function, the non-linear dynamical system in (18) mimics the Frank-Wolfe algorithm up to arbitrary precision and can therefore solve the kernel MEB problem.

From the point of view of neurocomputing this is of interest because, the system in (18) is algebraically equivalent to the equations that govern the internal dynamics of the simple recurrent architectures known as echo state networks [11]. In other words, we can think of this system in terms of a reservoir of n neurons whose synaptic connections are encoded in the matrix $2\mathbf{K}$. The system evolves with fixed inputs \mathbf{k} and its non-linear readout happens according to (6) and (7). The step size η_t assumes the role of the leaking rate of the reservoir. Since η_t decays towards zero, neural activities will stabilize and the system is guaranteed to approach a fixed point $\boldsymbol{\mu}_* = \lim_{t \rightarrow \infty} \boldsymbol{\mu}_t$.

What is further worth noting about the reservoir governed by (18) is that its synaptic connections and constant input are determined by the training data for the problem under consideration. Understanding the MEB problem as a learning task, both could be seen as a form of short term memory. At the beginning of a learning episode, data is loaded into this memory and used to determine support vectors. At the end of a learning episode, only those data points and activities required for decision making, i.e. those \mathbf{x}_i and μ_i for which $\mu_i > 0$, need to be persisted in a long term memory to be able to compute the characteristic function in (10).



(a) two moons



(b) three blobs

Fig. 2. Two additional 2D data sets. Squares highlight support vectors \mathbf{s}_j of Gaussian kernel MEBs; the color coding indicates the characteristic function $f(\mathbf{x})$ in (10). The panels on the right show local minima of $f(\mathbf{x})$. Points that minimize the function $f(\mathbf{x})$ can be understood as prototypes for the given data

4 Neural Reduction of Support Vectors to Prototypes

Figure 2 show two more 2D data sets for which we computed Gaussian kernel MEBs. Squares highlight support vectors, the coloring indicates values of the characteristic function $f(\mathbf{x})$ in (10), and blue dots represent its local minima. Both examples illustrate that (i) the number of support vectors of a kernel MEB is typically smaller than the number of data points the support vectors are computed from, (ii) the number of local minima of the characteristic function is typically smaller than the number of support vectors, and (iii) points where the characteristic function achieves a minimum constitute characteristic prototypes for the given data. Curiously, however, we are not aware of any prior work where minimizers of $f(\mathbf{x})$ have been considered as prototypes before. Next, we therefore discuss a simple recurrent procedure for how to compute them.

Solving the kernel MEB problem yields a vector of Lagrange multipliers whose non-zero entries indicate support vectors of \mathcal{B} . As the multipliers of all other data points equal zero, the characteristic function in (10) can be evaluated using only the support vectors and their multipliers.

Hence, letting $l \leq n$ denote the number of support vectors of \mathcal{B} , we next collect all of the support vectors of \mathcal{B} in a matrix $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_l] \in \mathbb{R}^{m \times l}$ and consider a vector $\boldsymbol{\sigma} \in \mathbb{R}^l$ of their multipliers. Furthermore, introducing a reduced

kernel matrix $\mathbf{Q} \in \mathbb{R}^{l \times l}$ where $Q_{ij} = K(\mathbf{s}_i, \mathbf{s}_j)$ and kernel vector $\mathbf{q} \in \mathbb{R}^l$ such that $\mathbf{q} = \text{diag}[\mathbf{Q}]$, allows us to rewrite the characteristic function in (10) as

$$f(\mathbf{x}) = \sqrt{K(\mathbf{x}, \mathbf{x}) - 2\boldsymbol{\kappa}^\top \boldsymbol{\sigma} + \boldsymbol{\sigma}^\top \mathbf{Q} \boldsymbol{\sigma}} - \sqrt{\boldsymbol{\sigma}^\top \mathbf{q} - \boldsymbol{\sigma}^\top \mathbf{Q} \boldsymbol{\sigma}} = \sqrt{d(\mathbf{x})} - r_* \quad (19)$$

where the entries of $\boldsymbol{\kappa} \in \mathbb{R}^l$ now amount to $\kappa_j = K(\mathbf{x}, \mathbf{s}_j)$ and where function $d : \mathbb{R}^m \rightarrow \mathbb{R}$ computes the squared feature space distance between \mathbf{x} and the center of \mathcal{B} .

Writing the characteristic function like this and observing that, on the outside of \mathcal{B} , the distance function $d(\mathbf{x})$ will grow beyond all bounds, it is clear that the problem of estimating local minimizers of $f(\mathbf{x})$ is equivalent to the problem of estimating those $\mathbf{x} \in \mathcal{B}$ for which the gradient of $d(\mathbf{x})$ vanishes.

Assuming that $K(\cdot, \cdot)$ is a Gaussian kernel such as in (9), we have $K(\mathbf{x}, \mathbf{x}) = 1$ so that the gradient of $d(\mathbf{x})$ becomes

$$\nabla d(\mathbf{x}) = -\frac{2}{\lambda^2} \sum_{j=1}^k \sigma_j K(\mathbf{x}, \mathbf{s}_j) [\mathbf{x} - \mathbf{s}_j]. \quad (20)$$

Equating the right hand side to $\mathbf{0}$ provides

$$\mathbf{x} = \frac{\sum_j \sigma_j K(\mathbf{x}, \mathbf{s}_j) \mathbf{s}_j}{\sum_j \sigma_j K(\mathbf{x}, \mathbf{s}_j)} = \frac{\sum_j \sigma_j \kappa_j \mathbf{s}_j}{\sum_j \sigma_j \kappa_j} = \frac{\mathbf{S} \boldsymbol{\Sigma} \boldsymbol{\kappa}}{\boldsymbol{\kappa}^\top \boldsymbol{\sigma}} = \mathbf{S} \boldsymbol{\Sigma} \mathbf{D}^{-1} \boldsymbol{\kappa} \quad (21)$$

where we introduced two diagonal matrices $\boldsymbol{\Sigma} = \text{diag}[\boldsymbol{\sigma}]$ and $\mathbf{D} = \text{diag}[(\boldsymbol{\sigma} \boldsymbol{\kappa}^\top) \mathbf{1}]$, respectively. But this result is to say that local minima of $f(\mathbf{x})$ correspond to weighted means or convex combinations of the support vectors of \mathcal{B} .

We also recognize (21) as an extension of classical mean shift updates [4, 10] (where there are no scaling parameters σ_j). Hence, when started with $\mathbf{x}_0 \in \mathbb{R}^m$, the following process with step size $\gamma_t \in [0, 1]$ will find the nearest minimizer of the characteristic function

$$\boldsymbol{\kappa}_t = \text{vec}[K(\mathbf{x}_t, \mathbf{s}_j)_j] \quad (22)$$

$$\mathbf{D}_t = \text{diag}[(\boldsymbol{\sigma} \boldsymbol{\kappa}_t^\top) \mathbf{1}] \quad (23)$$

$$\mathbf{x}_{t+1} = (1 - \gamma_t) \mathbf{x}_t + \gamma_t \mathbf{S} \boldsymbol{\Sigma} \mathbf{D}_t^{-1} \boldsymbol{\kappa}_t. \quad (24)$$

Looking at (24), we recognize these dynamics as yet another variant of the internal dynamics of a reservoir of neurons and note that, for $\gamma_t = 1$, the updates in (24) become the mean shift updates in (21). In other words, mode seeking via mean shifts can be seen as yet another form of neurocomputing.

Letting $\mathbf{x}_0 \leftarrow \mathbf{s}_j$ be a copy of one of the support vectors in \mathcal{S} and starting mode seeking at this point will identify the minimizer closest to this support vector. Repeating this process for all the support vectors of \mathcal{B} will thus collapse them into another, usually smaller, set of points that can be understood as prototypes of the given data. Collecting these in a matrix $\mathbf{P} \in \mathbb{R}^{m \times p}$ where $p \leq l \leq n$ therefore provides a reduced representation for a variety of downstream processing steps.

Table 1. Sample mean and prototypes extracted from the CBCL face data

\bar{x}	k -means prototypes	MEB prototypes
		

5 Practical Examples

In order to provide illustrative examples for the performance of our approach, we next present results obtained in experiments with three standard benchmark data sets: The MIT CBCL face database¹ contains intensity images of different faces recorded under various illumination conditions, the well known MNIST database [12] consist of intensity images of ten classes of handwritten digits, and the recently introduced MNIST-Fashion data [22] contains intensity images of fashion items again sampled from ten classes.

For each experimental setting, we vectorized the designated training samples which left us with a data matrix $\mathbf{X} \in \mathbb{R}^{361 \times 2429}$ for the CBCL data and matrices $\mathbf{X} \in \mathbb{R}^{361 \times 6000}$ for each class in two MNIST data sets.

In each experiment, we computed the sample mean $\bar{x} = \frac{1}{n} \mathbf{X} \mathbf{1}$ as a reference prototype and normalized the data in \mathbf{X} to zero mean and unit variance before running our procedure. Scale parameters λ for the Gaussian MEB kernels were determined using the method in [8] and reused during mean shift computation; the activation function for neural MEB computation was set to g_∞ .







An favorable property of MEB-based prototype identification is that it does not need manual specification of the number p of prototypes. Minimum enclosing ball computation and mean shift on the resulting support vectors automatically identify appropriate numbers l and p of support vectors and prototypes. Hence, after having obtained p MEB-based prototypes for each data matrix, we also ran k -means clustering for $k = p$ in order to provide an intuitive baseline comparison.

The rightmost column of Table 1 shows the $p = 29$ MEB-based prototypes we found for the CBCL face data; the center column of the table shows cluster prototypes resulting from k -means clustering for $k = 29$ and the single image in the leftmost column depicts the overall sample mean for comparison.

The cluster means in the center column represent average faces which are smoothed to an extent that makes it difficult to discern characteristic features. The MEB prototypes, on the other hand, show distinguishable and therefore

¹ <http://cbcl.mit.edu/software-datasets/FaceData2.html>.

Table 2. Sample mean and prototypes extracted from MNIST digit data


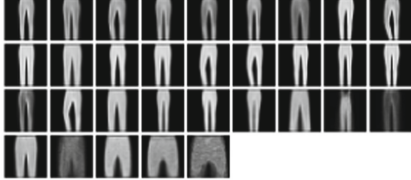
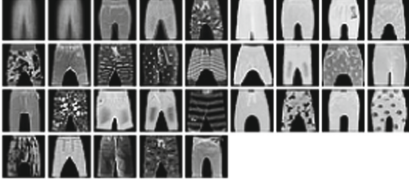

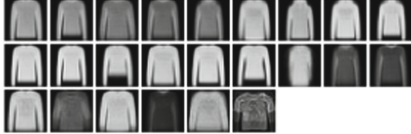





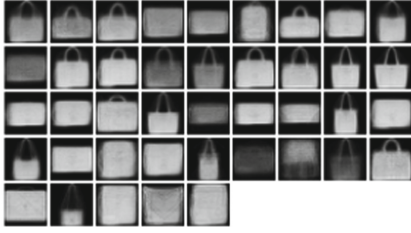

\bar{x}	k -means prototypes	MEB prototypes
		
		

interpretable visual characteristics. In other words, these prototypes reveal that the CBCL data contains pictures of faces of people of pale or dark complexion, of people wearing glasses, sporting mustaches, or having been photographed under varying illumination conditions.

What is further worth noting is that several of the MEB-based prototypes coincide with given images or, put differently, with actual data points. This phenomenon is known from latent factor models such as archetypal analysis [2, 5, 19] or CUR decompositions [13, 18, 21] and usually considered beneficial for interpretability [16]. The fact that we observe it here suggests that, for real world data, some of the support vectors of a kernel minimum enclosing ball themselves constitute minima of the corresponding characteristic function so that the above mean shift procedure will not reduce them any further. Since support vectors reside on the boundary of a given data set, this also explains the apparent variety among the MEB-based prototypes. While this also holds for prototypes extracted via archetypal analysis or CUR decompositions, the prototypes resulting from our approach do not exclusively coincide with extremal data points. In fact, some of them resemble the overall sample mean or the local means found via k -means. In contrast to archetypal analysis, CUR decompositions, or k -means clustering, we therefore observe that our MEB-based approach produces extremal *and* central prototypes simultaneously.

Tables 2 and 3 show examples of results obtained from the MNIST data sets. These are apparently analogous to the results we just discussed and therefore corroborate that our approach identifies prototypes that cover a wide variety of aspects of a data set.

Table 3. Sample mean and prototypes extracted from MNIST fashion data

\bar{x}	<i>k</i> -means prototypes	MEB prototypes
		
		
		
		

6 Conclusion

The problem of extracting representative prototypes from a given set of data frequently arises in data clustering, latent component analysis, manifold identification, or classifier training. Methods for this purpose should scale well and yield meaningful and interpretable results so as to assist downstream processing or decision making by analysts. In this paper, we proposed a two-stage approach based on kernel minimum enclosing balls and their characteristic functions. Our approach can be efficiently computed and empirical results suggest that it yields notably distinct prototypes that are therefore interpretable. Contrary to established techniques for clustering or factor analysis, our method yields central and extremal prototypes alike.

From the point of view of neurocomputing, our approach is interesting in that it can be computed using simple recurrent neural networks. Building on recent work in [1], we showed that kernel minimum enclosing balls can be computed using architectures akin to those found in reservoir computing. We also showed

that, if kernel minimum enclosing balls are determined w.r.t. Gaussian kernels, the problem of further reducing the support vectors of a ball naturally leads to a variant of the mean shift procedure which can be understood as a form of recurrent neural computation, too.

References

1. Bauckhage, C.: A neural network implementation of Frank-Wolfe optimization. In: Lintas, A., Rovetta, S., Verschure, P.F.M.J., Villa, A.E.P. (eds.) ICANN 2017. LNCS, vol. 10613, pp. 219–226. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68600-4_26
2. Bauckhage, C., Thureau, C.: Making archetypal analysis practical. In: Denzler, J., Notni, G., Süße, H. (eds.) DAGM 2009. LNCS, vol. 5748, pp. 272–281. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03798-6_28
3. Ben-Hur, A., Horn, D., Siegelmann, H., Vapnik, V.: Support vector clustering. *J. Mach. Learn. Res.* **2**(Dec), 125–137 (2001)
4. Cheng, Y.: Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **17**(8), 767–776 (1995). <https://doi.org/10.1109/34.400568>
5. Cutler, A., Breiman, L.: Archetypal analysis. *Technometrics* **36**(4), 338–347 (1994). <https://doi.org/10.1080/00401706.1994.10485840>
6. Dong, T., et al.: Imposing category trees onto word-embeddings using a geometric construction. In: Proceedings ICLR (2019)
7. Dong, T., Wang, Z., Li, J., Bauckhage, C., Cremers, A.: Triple classification using regions and fine-grained entity typing. In: Proceedings AAAI (2019)
8. Evangelista, P.F., Embrechts, M.J., Szymanski, B.K.: Some properties of the Gaussian kernel for one class learning. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 269–278. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74690-4_28
9. Frank, M., Wolfe, P.: An algorithm for quadratic programming. *Naval Res. Logist.* **3**(1–2), 95–110 (1956)
10. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function with applications in pattern recognition. *IEEE Trans. Inf. Theory* **21**(1), 32–40 (1975). <https://doi.org/10.1109/TIT.1975.1055330>
11. Jäger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* **304**(5667), 78–80 (2004). <https://doi.org/10.1126/science.1091277>
12. LeCun, Y., Boottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998). <https://doi.org/10.1109/5.726791>
13. Mahoney, M., Drineas, P.: CUR matrix decompositions for improved data analysis. *PNAS* **106**(3), 697–702 (2009). <https://doi.org/10.1073/pnas.0803205106>
14. Ruff, L., et al.: Deep one-class classification. In: Proceedings ICML (2018)
15. Schleif, F.M., Gisbrecht, A., Tino, P.: Supervised low rank indefinite kernel approximation using minimum enclosing balls. *Neurocomputing* **318**(Nov), 213–226 (2018). <https://doi.org/10.1016/j.neucom.2018.08.057>
16. Sifa, R.: An overview of Frank-Wolfe optimization for stochasticity constrained interpretable matrix and tensor factorization. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11140, pp. 369–379. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01421-6_36

17. Tax, D., Duin, R.: Support vector data description. *Mach. Learn.* **54**(1), 45–46 (2004). <https://doi.org/10.1023/B:MACH.0000008084.60811.49>
18. Thureau, C., Kersting, K., Bauckhage, C.: Deterministic CUR for improved large-scale data analysis: an empirical study. In: *Proceedings SDM*. SIAM (2012)
19. Thureau, C., Kersting, K., Wahabzada, M., Bauckhage, C.: Descriptive matrix factorization for sustainability: adopting the principle of opposites. *Data Min. Knowl. Discov.* **24**(2), 325–354 (2012). <https://doi.org/10.1007/s10618-011-0216-z>
20. Tsang, I., Kwok, J., Cheung, P.M.: Core vector machines: fast SVM training on very large data sets. *J. Mach. Learn. Res.* **6**(Apr), 363–392 (2010)
21. Wang, S., Zhang, Z.: Improving CUR matrix decompositions and the Nyström approximation via adaptive sampling. *J. Mach. Learn. Res.* **14**(1), 2729–2769 (2010)
22. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. [arXiv:1708.07747](https://arxiv.org/abs/1708.07747) [cs.LG] (2017)
23. Zhang, K., Kwok, J.: Clustered Nyström method for large scale manifold learning and dimension reduction. *IEEE Trans. Neural Netw.* **21**(10), 1576–1587 (2010). <https://doi.org/10.1109/TNN.2010.2064786>