Neurocomputing 74 (2011) 1372–1381

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Integrating feature maps and competitive layer architectures for motion segmentation

Jan Steffen [a], Michael Pardowitz [a,b], Jochen J. Steil [b], Helge Ritter [a,b,*]

[a] Neuroinformatics Group, Faculty of Technology, Bielefeld University, Germany
[b] Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, Germany

ARTICLE INFO

Available online 21 February 2011

Keywords:
Neural competition
Motion segmentation
Architecture
Structured manifolds
Unsupervised Kernel Regression
UKR
Competitive Layer Model
CLM

ABSTRACT

We present a generic approach to integrate feature maps with a competitive layer architecture to enable segmentation by a competitive neural dynamics specified in terms of the latent space mappings constructed by the feature maps. We demonstrate the underlying ideas for the case of motion segmentation, using a system that employs Unsupervised Kernel Regression (UKR) for the creation of the feature maps, and the Competitive Layer Model (CLM) for the competitive layer architecture. The UKR feature maps hold learned representations of a set of candidate motions and the CLM dynamics, working on features defined in the UKR domain, implements the segmentation of observed trajectory data according to the competing candidates. We also demonstrate how the introduction of an additional layer can provide the system with a parametrizable rejection mechanism for previously unknown observations. The evaluation on trajectories describing four different letters yields improved classification results compared to our previous, pure manifold approach.

1. Introduction

How can a neural system parse its complex, spatio-temporal input into more modular units that are sufficiently likely to re-occur to make them useful, yet specific enough to serve as building blocks for useful representations to shape the interaction behaviour of an agent with its environment?

Partial answers to this question have been found at various representational levels, leading to ideas for the adaptive formation of feature detectors, linear approaches such as PCA and ICA, non-linear models for creating topographic or dimension-reduced mappings, to competitive models for the dynamic decomposition of perceptual inputs into "Gestalt"-like entities motivated from the phenomenon of perceptual grouping.

Each of these models alone can only cover part of the entire parsing process towards meaningful high-level constituents. While classical, sequential computational approaches have been developed to a high level of sophistication for combining processing steps in a modular fashion into complex software systems, a similar level is currently largely missing for the combination of computations represented in a more parallel, "neural-style" format.

Advances along these lines will require to investigate generic schemes for combining a number of more elementary processing primitives into neurally motivated architectures, to explore their computational properties and how these can be shaped and connected with symbolic approaches in a problem-specific manner [1].

The present paper aims to contribute to that challenge by focusing on the task of motion segmentation for exploring the combination of two major approaches so far studied only in separation: (i) the creation of topology-preserving maps for creating compact, dimension-reduced representations and (ii) the use of a competitive and often layered dynamics utilizing a stack of feature maps for the decomposition of a complex pattern into more basic constituents by a Gestalt-like process.

We believe that the combination of the generic and by now well-researched processing primitives of map formation and dynamic pattern decomposition can be an important step towards the realization of more powerful architectures diminishing the gap between low-level input processing and the creation of higher-level representations.

Both processing primitives, map formation and dynamic pattern decomposition, could in principle be implemented by a variety of existing approaches. Our specific choice is biased by our own previous work, adopting Unsupervised Kernel Regression (UKR) as a representative method well-rooted in a statistical framework to create the map manifolds, and the Competitive Layer Model (CLM) as mathematically well-analysed recurrent network for the dynamic pattern decomposition step.

* Corresponding author at: Neuroinformatics Group, Faculty of Technology, Bielefeld University, Germany.
E-mail addresses: jsteil@cor-lab.uni-bielefeld.de (J.J. Steil), helge@techfak.uni-bielefeld.de (H. Ritter).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.11.028

With this point of departure, the plan of the paper is as follows: Sections 2 and 3 will provide a brief sketch of the UKR and CLM methods to make the paper more self-contained and to highlight the key aspects of both approaches that we deem synergistic for the proposed, integrated architecture. Section 4 then describes the architecture itself. Section 5 discusses how an extension of the CLM enables the proposed architecture to reject unrecognizable pattern elements. Section 6 introduces the data set used for our experiments, which themselves are evaluated in Section 7. Discussion and Conclusions are found in Section 8.

2. Unsupervised Kernel Regression (UKR)

Topology-preserving feature maps [2], originally motivated by attempts to model the formation of dimension-reduced latent variable representations in the brain, have stimulated the development of a range of successor methods aiming to implement the underlying key idea in computationally more efficient or statistically more principled ways.

Unsupervised Kernel Regression (*UKR*) is a recent example, introduced by Meinicke et al. [3,4] as the unsupervised counterpart of the Nadaraya–Watson kernel regression estimator. While a self-organizing map uses a set of "moveable pointers" to connect its map (the "latent") space with observation space locations, UKR replaces each moveable pointer by a kernel-weighted observation point. To make the resulting pointer moveable again, each observation point's kernel is now centred at a *moveable latent space location*. Thereby, the optimization of the mapping has now become the task of moving this set of pointers (kernel centres) *in the low-dimensional latent space*; moreover, the resulting mapping is no longer discretized but fully *continuous*.

More formally, given a discrete sample $\mathbf{Y}=(\mathbf{y}_1,\ldots,\mathbf{y}_N)\in\mathbb{R}^{d\times N}$ of d-dimensional observation points, the desired mapping $\mathbf{f}:\mathbf{x}\in\mathbb{R}^q\to\mathbf{y}\in\mathbb{R}^d$ from a low-dimensional, latent "map space" into the observation space is realized by a superposition

$$\mathbf{f}(\mathbf{x})=\sum_{i=1}^{N}\mathbf{y}_i\frac{K_{\boldsymbol{\beta}}(\mathbf{x}-\mathbf{x}_i)}{\sum_j K_{\boldsymbol{\beta}}(\mathbf{x}-\mathbf{x}_j)}$$

of kernels whose moveable centres (in the following denoted as $\mathbf{X}=(\mathbf{x}_1,\ldots,\mathbf{x}_N)\in\mathbb{R}^{q\times N}$, N denoting the size of the training set) are located in latent space. In UKR, $\mathbf{X}=\{x_i\}$ now plays the role of the regression parameters of the regression function and is treated as a set of *latent parameters* of a set of observed data $\mathbf{Y}=(\mathbf{y}_1,\ldots,\mathbf{y}_N)\in\mathbb{R}^{d\times N}$ and a corresponding functional relationship $\mathbf{y}=\mathbf{f}(\mathbf{x})$.

This distinguishes UKR from the usual kernel approaches, which consider an optimization of *linear* kernel *weights* plus a *choice* of a subset of kernel centres from *observation* points.

UKR training, i.e. finding optimal latent variables $\mathbf{X}$, is realized as gradient-based minimization of the reconstruction error $R(\mathbf{X})=(1/N)\sum_i\|\mathbf{y}_i-\mathbf{f}(\mathbf{x}_i;\mathbf{X})\|^2$. Since the scaling of the $\mathbf{x}_i$'s is only relevant relative to the kernel bandwidth $\boldsymbol{\beta}$, one can normalize the latter to $\boldsymbol{\beta}=1$.

Most notably, UKR can perform leave-one-out cross-validation without additional computational cost during the gradient descent [5]. In addition, it can easily be initialized with the results of spectral embedding methods like Isomap [6] or LLE [7] in order to improve its robustness against poor local minima.

By its construction, UKR shares with the "classical" self-organizing map [2] the absence of a direct representation for the inverse mapping $\mathbf{x}=\mathbf{f}^{-1}(\mathbf{y};\mathbf{X})$ from observation to latent space. Here, too, UKR requires a "bestmatch search" to identify the latent variable values that belong to an observation. For the case of UKR, this requires to solve the continuous minimization problem $\hat{\mathbf{x}}=\mathbf{g}(\mathbf{y};\mathbf{X})=\mathrm{argmin}_{\mathbf{x}}\|\mathbf{y}-\mathbf{f}(\mathbf{x};\mathbf{X})\|^2$ (cf. [3]).

In the original form, UKR is a purely unsupervised approach to continuous manifold learning. In order to incorporate prior knowledge about the structure of the training data, we introduced a structured version of UKR training (e.g. [8]). With structured UKR, it is possible to represent data with a temporal context, like trajectories of hand positions, in a very easy and robust way. In particular, due to the specific training of structured UKR, the order of the represented time series of training observations $\mathbf{y}_i$ is reflected in their latent parameters $\mathbf{x}_i$ and is captured by one specific latent time dimension. In order to represent periodic motions, we use the π-periodic kernel $K_{\circlearrowleft}(x_i-x_j;\beta)=\exp[-\frac{1}{2}\beta^2\sin^2(x_i-x_j)]$. For further details on UKR, refer to [3,4,8].

3. Competitive Layer Model (CLM)

With feature maps as a major generic processing unit in a neural system, a natural next step is to utilize a collection of feature maps to enable a structured decomposition of an observation into a set of "meaningful" constituents. A neurally plausible way to achieve such segmentation is to assume competitive neural dynamics, which have been investigated in several flavours in recent years [9–14].

The Competitive Layer Model (CLM, introduced in [15]) implements this idea in the form of a topographically structured activity competition among a stack of feature maps, leading to a flexible capability of pattern decomposition resembling Gestalt processes in perception [16,9,17]. Taking off from this basic version, it has been implemented in several ways [18,19] and so far investigated in a variety of contexts, ranging from image segmentation and perceptual grouping (e.g. [20–23,17]) to robot task learning [24,25].

These works so far have always used a stack of identical, and prespecified feature maps, although the CLM algorithm itself is not dependent on such a constraint. Before proceeding to the novel contribution of the present paper, the combination of the CLM decomposition approach with the data-driven construction of its layer feature maps by the UKR method, we give a brief sketch of the CLM approach.

A CLM consists of a set of layers (the feature maps), indexed by $\alpha=1..L$ and each containing a fixed number of neurons. We imagine this set as a vertical stack, denoting as "columns" the vertically arranged L-tuples of neurons that belong to the same lateral coordinate r in the stack, but to different layers α (see Fig. 1).

The activity of a neuron at position r within layer α is a non-negative value denoted by $x_{r\alpha}$ (not to be confused with the latent variables $\mathbf{x}_r$ introduced in the previous section and distinguishable in our notation only from the absence of the α index!). An input pattern is a spatial activity pattern h_r associated with a (fixed) collection of feature vectors $\mathbf{m}_r$ attached to the coordinate locations r of the columns.

Within each column, its maximally responding neuron determines which layer is "assigned" by the CLM to the current input activity h_r at the column's location.

To shape the neural dynamics to result in "meaningful" assignments of the input activities to layers (and thereby to group them), the neuron responses $x_{r\alpha}$ are subjected to three different interactions (cf. Fig. 1):

The first two interactions connect only neurons within the same column ("vertical" interactions):

- a linear excitation from inputs h_r to all neurons $x_{r\alpha}$ of the same column at r, and
- an inhibitory interaction within the column at r trying to keep the columnar activity sum $\sum_\alpha x_{r\alpha}$ proportional to the column's input h_r.