Mach Learn (2017) 106:1359–1380
DOI 10.1007/s10994-017-5632-x

Graph-based predictable feature analysis

Björn Weghenkel[1] · **Asja Fischer**[2] · **Laurenz Wiskott**[1]

Received: 22 December 2015 / Accepted: 1 March 2017 / Published online: 9 May 2017

Abstract We propose graph-based predictable feature analysis (GPFA), a new method for unsupervised learning of predictable features from high-dimensional time series, where high predictability is understood very generically as low variance in the distribution of the next data point given the previous ones. We show how this measure of predictability can be understood in terms of graph embedding as well as how it relates to the information-theoretic measure of predictive information in special cases. We confirm the effectiveness of GPFA on different datasets, comparing it to three existing algorithms with similar objectives—namely slow feature analysis, forecastable component analysis, and predictable feature analysis—to which GPFA shows very competitive results.

Keywords Unsupervised learning · Dimensionality reduction · Feature learning · Representation learning · Graph embedding · Predictability

1 Introduction

When we consider the problem of an agent (artificial or biological) interacting with its environment, its signal processing is naturally embedded in time. In such a scenario, a feature's ability to predict the future is a necessary condition for it to be useful in any behaviorally rel-

Editors: Thomas Gärtner, Mirco Nanni, Andrea Passerini, and Celine Robardet.

✉ Björn Weghenkel
bjoern.weghenkel@rub.de

Asja Fischer
fischera@iai.uni-bonn.de

Laurenz Wiskott
laurenz.wiskott@rub.de

[1] Institute for Neural Computation, Ruhr University Bochum, Bochum, Germany

[2] Computer Science Institute, University of Bonn, Bonn, Germany

evant way: A feature that does not hold information about the future is out-dated the moment it is processed and any action based on such a feature can only be expected to have random effects.

As an practical example, consider a robot interacting with its environment. When its stream of sensory input is high-dimensional (e.g., the pixel values from a camera), we are interested in mapping this input to a lower-dimensional representation to make subsequent machine learning steps and decision making more robust and efficient. At this point, however, it is crucial not to throw away information that the input stream holds about the future as any subsequent decision making will depend on this information. The same holds for time series like video, weather, or business data: when performing classification or regression on the learned features, or when the data is modelled for instance by a (hidden) Markov model, we are mostly interested in features that have some kind of predictive power.

Standard algorithms for dimensionality reduction (DR), like PCA, however, are designed to preserve properties of the data that are not (or at least not explicitly) related to predictability and thus are likely to waste valuable information that could be extracted from the data's temporal structure. In this paper we will therefore focus on the unsupervised learning of predictable features for high-dimensional time series, that is, given a sequence of data points in a high-dimensional vector space we are looking for the projection into a sub-space which makes predictions about the future most reliable.

While aspects of predictability are (implicitly) dealt with through many different approaches in machine learning, only few algorithms have addressed this problem of finding subspaces for multivariate time series suited for predicting the future. The recently proposed *forecastable component analysis* (ForeCA) (Goerg 2013) is based on the idea that predictable signals can be recognized by their low entropy in the power spectrum while white noise in contrast would result in a power spectrum with maximal entropy. *Predictable feature analysis* (PFA) (Richthofer and Wiskott 2013) focuses on signals that are well predictable through autoregressive processes. Another DR approach that was not designed to extract predictable features but explicitly takes into account the temporal structure of the data is *slow feature analysis* (SFA) (Wiskott and Sejnowski 2002). Still, the resulting slow features can be seen as a special case of predictable features (Creutzig and Sprekeler 2008). For reinforcement learning settings, *predictive projections* (Sprague 2009) and *robotic priors* (Jonschkowski and Brock 2015) learn mappings where actions applied to similar states result in similar successor states. Also, there are recent variants of PCA that at least allow for weak statistical dependence between samples (Han and Liu 2013).

All in all, however, the field of unsupervised learning of predictable subspaces for time series is largely unexplored. Our contribution consists of a new measure of the predictability of learned features as well as of an algorithm for learning those. The proposed measure has the advantage of being very generic, of making only few assumptions about the data at hand, and of being easy to link to the information-theoretic quantity of *predictive information* (Bialek et al. 2001), that is, the mutual information between past and future. The proposed algorithm, *graph-based predictable feature analysis* (GPFA), not only shows very competitive results in practice but also has the advantage of being very flexible, and of allowing for a variety of future extensions. Through its formulation in terms of a graph embedding problem, it can be straightforwardly combined with many other, mainly geometrically motivated objectives that have been formulated in the graph embedding framework (Yan et al. 2007)—like Isomap (Tenenbaum et al. 2000), Locally Linear Embedding (LLE, Roweis and Saul 2000), Laplacian Eigenmaps (Belkin and Niyogi 2003), and Locality Preserving Projections (LPP, He and Niyogi 2004). Moreover, GPFA could make use of potential speed-ups like spectral regression (Cai et al. 2007), include additional label information in its graph like in (Escalante et al.

2013), or could be applied to non-vectorial data like text. Kernelization and other approaches to use GPFA in a non-linear way are discussed in Sect. 5.

The remaining paper is structured as follows. In Sect. 2 we derive the GPFA algorithm. We start by introducing a new measure of predictability (Sect. 2.1), a consistent estimate for it (Sect. 2.2), and a simplified version of the estimate which is used by the proposed algorithm as an intermediate step (Sect. 2.3). Then the link to the graph embedding framework is established in Sects. 2.4 and 2.5. After describing three useful heuristics in Sect. 2.6, the core algorithm is summarized in Sect. 2.7 and an iterated version of the algorithm is described in Sect. 2.8. Afterwards the algorithm is analyzed with respect to its objective's close relation to predictive information (Sect. 2.9) and with respect to its time complexity (Sect. 2.10). Section 3 summarizes the most closely related approaches for predictable feature learning—namely SFA, ForeCA, and PFA—and Sect. 4 describes experiments on different datasets. We end with a discussion of limitations, open questions and ideas which shall be conducted by future research in Sect. 5 and with a conclusion in Sect. 6.

2 Graph-based predictable feature analysis

Given is a time series $\mathbf{x}_t \in \mathbb{R}^N$, $t = 1, \ldots, S$, as training data that is assumed to be generated by a stationary stochastic process $(\boldsymbol{X}_t)_t$ of order p. The goal of GPFA is to find a lower-dimensional feature space for that process by means of an orthogonal transformation $\mathbf{A} \in \mathbb{R}^{N \times M}$, leading to projected random variables $\boldsymbol{Y}_t = \mathbf{A}^T \boldsymbol{X}_t$ with low average variance given the state of the p previous time steps. We use $\boldsymbol{X}_t^{(p)}$ to denote the concatenation $(\boldsymbol{X}_t^T, \ldots, \boldsymbol{X}_{t-p+1}^T)^T$ of the p predecessor of $\boldsymbol{X}_{t+1}$ to simplify notation. The corresponding state values are vectors in $\mathbb{R}^{N \cdot p}$ and denoted by $\mathbf{x}_t^{(p)}$.

2.1 Measuring predictability

We understand the predictability of the learned feature space in terms of the variance of the projected random variables $\boldsymbol{Y}_t$ in this space: the lower their average variance given their p-step past, the higher the predictability. We measure this through the expected covariance matrix of $\boldsymbol{Y}_{t+1}$ given $\boldsymbol{Y}_t^{(p)}$ and minimize it in terms of its trace, i.e., we minimize the sum of variances in all principal directions. Formally, we look for the projection matrix $\mathbf{A}$ leading to a projected stochastic process $(\boldsymbol{Y}_t)_t$ with minimum

$$\operatorname{trace}\left(\mathbb{E}_{\boldsymbol{Y}_t^{(p)}}\left[\operatorname{cov}\left(\boldsymbol{Y}_{t+1} | \boldsymbol{Y}_t^{(p)}\right)\right]\right). \tag{1}$$

For simplicity, we refer to this as "minimizing the variance" in the following. When we make the generally reasonable assumption of $p(\boldsymbol{Y}_{t+1} | \boldsymbol{Y}_t^{(p)} = \mathbf{y}_t^{(p)})$ being Gaussian, that makes the learned features a perfect fit to be used in combination with least-squares prediction models.[1] For non-Gaussian conditional distributions we assume the variance to function as a useful proxy for quantifying the uncertainty of the next step. Note, however, that assuming Gaussianity for the conditional distributions does not imply or require Gaussianity of $\boldsymbol{X}_t$ or of the joint distributions $p(\boldsymbol{X}_s, \boldsymbol{X}_t)$, $s \neq t$, which makes the predictability measure applicable to a wide range of stochastic processes.

[1] Note that in the Gaussian case the covariance not only covers the distribution's second moments but is sufficient to describe the higher-order moments as well.

2.2 Estimating predictability

In practice, the expected value in (1) can be estimated by sampling a time series $\mathbf{y}_1, \ldots, \mathbf{y}_S$ from the process $(\boldsymbol{Y}_t)_t$. However, the empirical estimate for the covariance matrices $\mathrm{cov}(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)} = \mathbf{y}_t^{(p)})$, with $\mathbf{y}_t^{(p)} \in \mathbb{R}^{M \cdot p}$, is not directly available because there might be only one sample of $\boldsymbol{Y}_{t+1}^{(p)}$ with previous state value $\mathbf{y}_t^{(p)}$. Therefore we calculate a k-nearest neighbor (kNN) estimate instead. Intuitively, the sample size is increased by also considering the k points that are most similar (e.g., in terms of Euclidean distance) to $\mathbf{y}_t^{(p)}$, assuming that a distribution $p(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)} = \mathbf{y}'^{(p)}_t)$ is similar to $p(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)} = \mathbf{y}_t^{(p)})$ if $\mathbf{y}'^{(p)}_t$ is close to $\mathbf{y}_t^{(p)}$. In other words, we group together signals that are similar in their past p steps. To that end, a set $\mathcal{K}_t^{(p)}$ is constructed, containing the indices of all k nearest neighbors of $\mathbf{y}_t^{(p)}$ (plus the 0-st neighbor, t itself), i.e., $\mathcal{K}_t^{(p)} := \{i \mid \mathbf{y}_i^{(p)} \text{ is kNN of } \mathbf{y}_t^{(p)}, i = 1, \ldots, S\} \cup \{t\}$. The covariance is finally estimated based on the successors of these neighbors. Formally, the k-nearest neighbor estimate of (1) is given by

$$\mathrm{trace}\left(\left\langle \mathrm{cov}\left(\left\{\mathbf{y}_{i+1} \mid i \in \mathcal{K}_t^{(p)}\right\}\right)\right\rangle_t\right), \tag{2}$$

where $\langle\cdot\rangle_t$ denotes the average over t. Note that the distance measure used for the k nearest neighbors does not necessarily need to be Euclidean. Think for instance of "perceived similarities" of words or faces.

While we introduce the kNN estimate here to assess the uncertainty inherent in the stochastic process, we note that it may be of practical use in a deterministic setting as well. For a deterministic dynamical system the kNN estimate includes nearby points belonging to nearby trajectories in the dataset. Thus, the resulting feature space may be understood as one with small divergence of neighboring trajectories (as measured through the Lyapunov exponent, for instance).

2.3 Simplifying predictability

Finding the transformation $\mathbf{A}$ that leads to the most predictable $(\boldsymbol{Y}_t)_t$ in the sense of (1) becomes difficult through the circumstance that the predictability can only be evaluated *after* $\mathbf{A}$ has been fixed. The circular nature of this optimization problem motivates the iterated algorithm described in Sect. 2.8. As a helpful intermediate step we define a weaker measure of predictability that is conditioned on the input $\boldsymbol{X}_t$ instead of the features $\boldsymbol{Y}_t$ and has a closed-form solution, namely minimizing

$$\mathrm{trace}\left(\mathbb{E}_{\boldsymbol{X}_t^{(p)}}\left[\mathrm{cov}\left(\boldsymbol{Y}_{t+1}|\boldsymbol{X}_t^{(p)}\right)\right]\right)$$

via its k-nearest neighbor estimate

$$\mathrm{trace}\left(\left\langle \mathrm{cov}\left(\left\{\mathbf{y}_{i+1} \mid i \in \tilde{\mathcal{K}}_t^{(p)}\right\}\right)\right\rangle_t\right). \tag{3}$$

Analogous to $\mathcal{K}_t^{(p)}$, the set $\tilde{\mathcal{K}}_t^{(p)}$ contains the indices of the k nearest neighbors of $\mathbf{x}_t^{(p)}$ plus t itself. Under certain mild mixing assumptions for the stochastic process, the text-book results on k-nearest neighbor estimates can be applied to auto-regressive time series as well (Collomb 1985). Thus, in the limit of $S \to \infty, k \to \infty, k/S \to 0$, the estimated covariance

$$\mathrm{cov}\left(\left\{\mathbf{y}_{i+1} \mid i \in \tilde{\mathcal{K}}_t^{(p)}\right\}\right) = \left\langle \mathbf{y}_{i+1}\mathbf{y}_{i+1}^T\right\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}} - \left\langle \mathbf{y}_{i+1}\right\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}\left\langle \mathbf{y}_{i+1}\right\rangle^T_{i\in\tilde{\mathcal{K}}_t^{(p)}}$$

converges to

$$\mathbb{E}\left[\boldsymbol{Y}_{t+1}\boldsymbol{Y}_{t+1}^T|\boldsymbol{X}_t^{(p)}=\mathbf{x}_t^{(p)}\right]-\mathbb{E}\left[\boldsymbol{Y}_{t+1}|\boldsymbol{X}_t^{(p)}=\mathbf{x}_t^{(p)}\right]\mathbb{E}\left[\boldsymbol{Y}_{t+1}|\boldsymbol{X}_t^{(p)}=\mathbf{x}_t^{(p)}\right]^T,$$

i.e., it is a consistent estimator of $\mathrm{cov}(\boldsymbol{Y}_{t+1}|\boldsymbol{X}_t^{(p)}=\mathbf{x}_t^{(p)})$.

When measuring predictability, one assumption made about the process $(\boldsymbol{X}_t)_t$ in the following is that it is already white, i.e., $\mathbb{E}[\boldsymbol{X}_t]=\mathbf{0}$ and $\mathrm{cov}(\boldsymbol{X}_t)=\mathbf{I}$ for all t. Otherwise components with lower variance would tend to have higher predictability *per se*.

2.4 Predictability as graph

Instead of optimizing objective (3) directly, we reformulate it such that it can be interpreted as the embedding of an undirected graph on the set of training samples. Consider the graph to be represented by a symmetric connection matrix $\mathbf{W}=(\mathrm{W}_{ij})_{ij}\in\mathbb{R}^{S\times S}$ with weights $\mathrm{W}_{ij}=\mathrm{W}_{ji}>0$ whenever two nodes corresponding to vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ from the training sequence are connected by an edge $\{\mathbf{x}_i,\mathbf{x}_j\}$. Further assume an orthogonal transformation $\mathbf{A}\in\mathbb{R}^{N\times M}$ for that graph with $M\ll N$ that minimizes

$$\sum_{i,j=1}^{S}\mathrm{W}_{ij}\|\mathbf{A}^T\mathbf{x}_i-\mathbf{A}^T\mathbf{x}_j\|^2=\sum_{i,j=1}^{S}\mathrm{W}_{ij}\|\mathbf{y}_i-\mathbf{y}\|^2. \tag{4}$$

Intuitively, this term becomes small if the projections of points connected in the graph (i.e., nodes for which $\mathrm{W}_{ij}>0$) are close to each other, while there is no penalty for placing the projections of unconnected points far apart.

Through a proper selection of the weights W_{ij}, the transformation $\mathbf{A}$ can be used to maximize predictability in the sense of minimizing (3). This becomes clear by noting that the trace of the sample covariance

$$\begin{aligned}\mathrm{cov}\left(\left\{\mathbf{y}_{i+1}\mid i\in\tilde{\mathcal{K}}_t^{(p)}\right\}\right)&=\left\langle\mathbf{y}_{i+1}\mathbf{y}_{i+1}^T\right\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}-\langle\mathbf{y}_{i+1}\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}\langle\mathbf{y}_{i+1}\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}^T\\&=\left\langle(\mathbf{y}_{i+1}-\overline{\mathbf{y}_{i+1}})(\mathbf{y}_{i+1}-\overline{\mathbf{y}_{i+1}})^T\right\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}},\end{aligned}$$

with $\overline{\mathbf{y}_{i+1}}=\langle\mathbf{y}_{i+1}\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}$ being the sample mean, can always be formulated via pairwise differences of samples, since

$$\begin{aligned}&\mathrm{trace}\left(\left\langle(\mathbf{y}_{i+1}-\overline{\mathbf{y}_{i+1}})(\mathbf{y}_{i+1}-\overline{\mathbf{y}_{i+1}})^T\right\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}\right)=\left\langle(\mathbf{y}_{i+1}-\overline{\mathbf{y}_{i+1}})^T(\mathbf{y}_{i+1}-\overline{\mathbf{y}_{i+1}})\right\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}\\&=\left\langle\mathbf{y}_{i+1}^T\mathbf{y}_{i+1}\right\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}-\langle\mathbf{y}_{i+1}\rangle_{i\in\tilde{\mathcal{K}}_t^{(p)}}^T\langle\mathbf{y}_{j+1}\rangle_{j\in\tilde{\mathcal{K}}_t^{(p)}}=\left\langle\mathbf{y}_{i+1}^T\mathbf{y}_{i+1}-\mathbf{y}_{i+1}^T\mathbf{y}_{j+1}\right\rangle_{i,j\in\tilde{\mathcal{K}}_t^{(p)}}\\&=\frac{1}{2}\left\langle\mathbf{y}_{i+1}^T\mathbf{y}_{i+1}-2\mathbf{y}_{i+1}^T\mathbf{y}_{j+1}+\mathbf{y}_{j+1}^T\mathbf{y}_{j+1}\right\rangle_{i,j\in\tilde{\mathcal{K}}_t^{(p)}}=\frac{1}{2}\left\langle\|\mathbf{y}_{i+1}-\mathbf{y}_{j+1}\|^2\right\rangle_{i,j\in\tilde{\mathcal{K}}_t^{(p)}}.\end{aligned} \tag{5}$$

Thus, by incrementing weights[2] of the edges $\{\mathbf{y}_{i+1},\mathbf{y}_{j+1}\}$ for all $i,j\in\tilde{\mathcal{K}}_t^{(p)}, t=p,\ldots,S-1$, minimizing (4) directly leads to the minimization of (3).

Note that for the construction of the graph, the data actually does not need to be represented by points in a vector space. Data points also could, for instance, be words from a text corpus as long as there are either enough samples per word or there is an applicable distance measure to determine "neighboring words" for the k-nearest neighbor estimates.

[2] All edge weights are initialized with zero.

2.5 Graph embedding

To find the orthogonal transformation $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_M) \in \mathbb{R}^{N\times M}$ that minimizes (4), let the training data be concatenated in $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_S) \in \mathbb{R}^{N\times S}$, and let $\mathbf{D} \in \mathbb{R}^{S\times S}$ be a diagonal matrix with $\mathrm{D}_{ii} = \sum_j \mathrm{W}_{ij}$ being the sum of edge weights connected to node $\mathbf{x}_i$. Let further $\mathbf{L} := \mathbf{D} - \mathbf{W}$ be the graph Laplacian. Then, the minimization of (4) can be re-formulated as a minimization of

$$\begin{aligned}
&\frac{1}{2}\sum_{i,j=1}^{S} \mathrm{W}_{ij}\|\mathbf{A}^T\mathbf{x}_i - \mathbf{A}^T\mathbf{x}_j\|^2 \\
&= \frac{1}{2}\sum_{i,j=1}^{S} \mathrm{W}_{ij}\,\mathrm{trace}\left(\left(\mathbf{A}^T\mathbf{x}_i - \mathbf{A}^T\mathbf{x}_j\right)\left(\mathbf{A}^T\mathbf{x}_i - \mathbf{A}^T\mathbf{x}_j\right)^T\right) \\
&= \mathrm{trace}\left(\sum_{i=1}^{S}\mathbf{A}^T\mathbf{x}_i\mathrm{D}_{ii}\mathbf{x}_i^T\mathbf{A} - \sum_{i,j=1}^{S}\mathbf{A}^T\mathbf{x}_i\mathrm{W}_{ij}\mathbf{x}_j^T\mathbf{A}\right) \\
&= \mathrm{trace}(\mathbf{A}^T\mathbf{X}(\mathbf{D}-\mathbf{W})\mathbf{X}^T\mathbf{A}) = \mathrm{trace}(\mathbf{A}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{A}) = \sum_{i=1}^{M}\mathbf{a}_i^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{a}_i\,. \qquad (6)
\end{aligned}$$

The $\mathbf{a}_i$ that minimize (6) are given by the first ("smallest") M eigenvectors of the eigenvalue problem

$$\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{a} = \lambda\mathbf{a}. \qquad (7)$$

See He and Niyogi (2004) for the analogous derivation of the one-dimensional case that was largely adopted here as well as for a kernelized version of the graph embedding.

2.6 Additional heuristics

The following three heuristics proved to be useful for improving the results in practice.

Normalized graph embedding

First, in the context of graph embedding, the minimization of $\mathbf{a}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{a}$ described in the section above is often solved subject to the additional constraint $\mathbf{a}^T\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{a} = 1$ (see for instance He and Niyogi 2004; Luxburg 2007). Through this constraint the projected data points are normalized with respect to their degree of connectivity in every component $\mathbf{Y} = \mathbf{X}^T\mathbf{a}$, i.e., $\mathbf{Y}^T\mathbf{D}\mathbf{Y} = 1$. Objective function and constraint can be combined in the Lagrange function $\mathbf{a}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{a} - \lambda(\mathbf{a}^T\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{a} - 1)$. Then the solution is given by the "smallest" eigenvectors of the generalized eigenvalue problem

$$\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{a} = \lambda\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{a}. \qquad (8)$$

Solving this generalized eigenvalue problem instead of (7) tended to improve the results for GPFA.

Minimizing variance of the past

Second, while not being directly linked to the above measure of predictability, results benefit significantly when the variance of the past is minimized simultaneously to that of the future.

To be precise, additional edges $\{\mathbf{y}_{i-p}, \mathbf{y}_{j-p}\}$ are added to the graph for all $i, j \in \tilde{\mathcal{K}}_t^{(p)}$, $t = p+1, \ldots, S$. The proposed edges here have the effect of mapping states with *similar* futures to *similar* locations in feature space. In other words, states are represented with respect to what is expected in the next steps (not with respect to their past). Conceptually this is related to the idea of *causal states* (Shalizi and Crutchfield 2001), where all (discrete) states that share the same conditional distribution over possible futures are mapped to the same causal state (also see Still (2009) for a closely related formulation in interactive settings).

Star-like graph structure

As a third heuristic, the graph above can be simplified by replacing the sample mean $\overline{\mathbf{y}_{i+1}}$ in the minimization objective

$$\begin{aligned}
&\text{trace}\left(\left\langle \text{cov}\left(\left\{\mathbf{y}_{i+1} \mid i \in \tilde{\mathcal{K}}_t^{(p)}\right\}\right)\right\rangle_t\right) \\
&= \text{trace}\left(\left\langle (\mathbf{y}_{i+1} - \overline{\mathbf{y}_{i+1}})(\mathbf{y}_{i+1} - \overline{\mathbf{y}_{i+1}})^T \right\rangle_{i \in \tilde{\mathcal{K}}_t^{(p)}}\right) \\
&= \left\langle \|\mathbf{y}_{i+1} - \overline{\mathbf{y}_{i+1}}\|^2 \right\rangle_{i \in \tilde{\mathcal{K}}_t^{(p)}}
\end{aligned}$$

by $\mathbf{y}_{t+1}$. This leads to

$$\left\langle \|\mathbf{y}_{i+1} - \mathbf{y}_{t+1}\|^2 \right\rangle_{i \in \tilde{\mathcal{K}}_t^{(p)}}, \tag{9}$$

inducing a graph with star-like structures. It is constructed by adding (undirected) edges $\{\mathbf{y}_{i+1}, \mathbf{y}_{t+1}\}$ for all $i \in \tilde{\mathcal{K}}_t^{(p)}$. Analogously, edges for reducing the variance of the past are given by $\{\mathbf{y}_{i-p}, \mathbf{y}_{t-p}\}$ for $i \in \tilde{\mathcal{K}}_t^{(p)}$.

We refer to the resulting algorithms as GPFA (1) and GPFA (2), corresponding to the graphs defined through (5) and (9), respectively. See Fig. 1 for an illustration of both graphs. The differences in performance are empirically evaluated in Sect. 4.

2.7 Algorithm

In the following, the core algorithm is summarized step by step, where training data $\mathbf{x}_1, \ldots, \mathbf{x}_S$ is assumed to be white already or preprocessed accordingly (in that case, the same transformation has to be taken into account during subsequent feature extractions). Lines starting with (1) and (2) indicate the steps for GPFA (1) and GPFA (2), respectively.

1. **Calculate neighborhood**
   For every $\mathbf{x}_t^{(p)}$, $t = p, \ldots, S$, calculate index set $\tilde{\mathcal{K}}_t^{(p)}$ of k nearest neighbors (plus t itself).
2. **Construct graph (future)**
   Initialize connection matrix $\mathbf{W}$ to zero. For every $t = p, \ldots, S-1$, add edges, according to either
   (1) $\mathrm{W}_{i+1,j+1} \leftarrow \mathrm{W}_{i+1,j+1} + 1 \quad \forall i, j \in \tilde{\mathcal{K}}_t^{(p)}$ or
   (2) $\mathrm{W}_{i+1,t+1} \leftarrow \mathrm{W}_{i+1,t+1} + 1$ and $\mathrm{W}_{t+1,i+1} \leftarrow \mathrm{W}_{t+1,i+1} + 1 \quad \forall i \in \tilde{\mathcal{K}}_t^{(p)} \setminus \{t\}$.
3. **Construct graph (past)**
   For every $t = p+1, \ldots, S$, add edges, according to either
   (1) $\mathrm{W}_{i-p,j-p} \leftarrow \mathrm{W}_{i-p,j-p} + 1 \quad \forall i, j \in \tilde{\mathcal{K}}_t^{(p)}$ or
   (2) $\mathrm{W}_{i-p,t-p} \leftarrow \mathrm{W}_{i-p,t-p} + 1$ and $\mathrm{W}_{t-p,i-p} \leftarrow \mathrm{W}_{t-p,i-p} + 1 \quad \forall i \in \tilde{\mathcal{K}}_t^{(p)} \setminus \{t\}$.

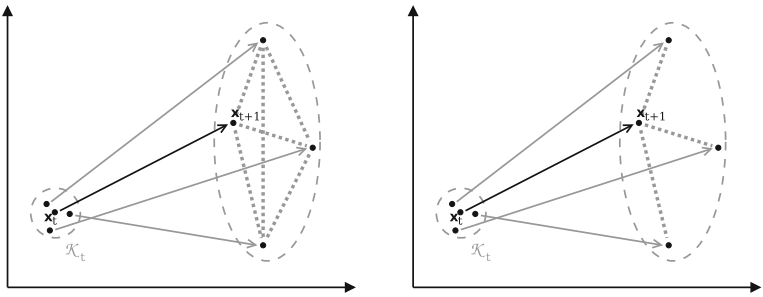

Fig. 1 Graphs constructed for GPFA (1) and GPFA (2) are illustrated on the *left* and *right*, respectively. Both pictures show a sample $\mathbf{x}_t$ and its k nearest neighbors together with their successors in time (indicated through *arrows*). The distribution of the successors indicates that the first axis can be predicted with less uncertainty than the second axis. The *dotted lines* depict edges that are added to the graph according to the two variants of the algorithm. Edges for minimizing the variance of the past are constructed analogously

4. **Linear graph embedding**
   Calculate $\mathbf{L}$ and $\mathbf{D}$ as defined in Sect. 2.5.

   Find the first ("smallest") M solutions to $\mathbf{XLX}^T\mathbf{a} = \lambda\mathbf{XDX}^T\mathbf{a}$ and normalize them, i.e., $||\mathbf{a}|| = 1$.

2.8 Iterated GPFA

As shown in Sect. 2.4, the core algorithm above produces features $(\boldsymbol{Y}_t)_t$ with low trace $(\mathbb{E}_{\boldsymbol{X}_t^{(p)}}[\mathrm{cov}(\boldsymbol{Y}_{t+1}|\boldsymbol{X}_t^{(p)})])$. In many cases these features may already be predictable in themselves, that is, they have a low trace $(\mathbb{E}_{\boldsymbol{Y}_t^{(p)}}[\mathrm{cov}(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)})])$. There are, however, cases where the results of both objectives can differ significantly (see Fig. 2 for an example of such a case). Also, the k-nearest neighbor estimates of the covariances become increasingly unreliable in higher-dimensional spaces.

Therefore, we propose an iterated version of the core algorithm as a heuristic to address these problems. First, an approximation of the desired covariances $\mathrm{cov}(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)} = \mathbf{y}_t^{(p)})$ can be achieved by rebuilding the graph according to neighbors of $\mathbf{y}_t^{(p)}$, not $\mathbf{x}_t^{(p)}$. This in turn may change the whole optimization problem, which is the reason to repeat the whole procedure several times. Second, calculating the sample covariance matrices based on the k nearest neighbors of $\mathbf{y}_t^{(p)} \in \mathbb{R}^{M\cdot p}$ instead of $\mathbf{x}_t^{(p)} \in \mathbb{R}^{N\cdot p}$ counteracts the problem of unreliable k-nearest neighbor estimates in high-dimensional spaces, since $M \cdot p \ll N \cdot p$.

The resulting (iterated) GPFA algorithm works like this:

(a) Calculate neighborhoods $\tilde{\mathcal{K}}_t^{(p)}$ of $\mathbf{x}_t^{(p)}$ for $t = p, \ldots, S-1$.
(b) Perform steps 2–4 of GPFA as described in Sect. 2.7.
(c) Calculate projections $\mathbf{y}_t = \mathbf{A}^T\mathbf{x}_t$ for $t = 1, \ldots, S$.
(d) Calculate neighborhoods[3] $\mathcal{K}_t^{(p)}$ of $\mathbf{y}_t^{(p)}$ for $t = p, \ldots, S-1$.
(e) Start from step (b), using $\mathcal{K}_t^{(p)}$ instead of $\tilde{\mathcal{K}}_t^{(p)}$.

where steps (b)–(e) are either repeated for R iterations or until convergence.

[3] Of course, this step is not necessary for the last iteration.

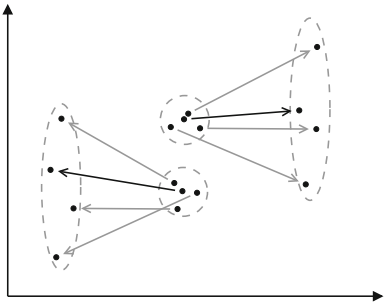

Fig. 2 Schematic illustration of a case where $p(\mathbf{y}_{t+1}|\boldsymbol{X}_t = \mathbf{x}_t)$ and $p(\mathbf{y}_{t+1}|\boldsymbol{Y}_t = \mathbf{y}_t)$ differ significantly. Points from two neighborhoods are shown together with their immediate successors in time. The distributions of the successors indicate that the first axis would be the most predictable direction. However, projecting all points on the first axis would result in a feature that is highly unpredictable. Therefore another direction will likely be preferred in the next iteration

While we can not provide a theoretical guarantee for the iterative process to converge, it did so in practice in all of our experiments (see Sect. 4). Also note that in general there is no need for the dimensionality M of the intermediate projections $\mathbf{y}_t \in \mathbb{R}^M$ to be the same as for the final feature space.

2.9 Relationship to predictive information

Predictive information—that is, the mutual information between past states and future states—has been used as a natural measure of how well-predictable a stochastic process is (e.g., Bialek and Tishby 1999; Shalizi and Crutchfield 2001). In this section we discuss under which conditions the objective of GPFA corresponds to extracting features with maximal predictive information.

Consider again the stationary stochastic process $(\boldsymbol{X}_t)_t$ of order p and its extracted features $\boldsymbol{Y}_t = \mathbf{A}^T \boldsymbol{X}_t$. Their predictive information is given by

$$I\left(\boldsymbol{Y}_{t+1}; \boldsymbol{Y}_t^{(p)}\right) = H(\boldsymbol{Y}_{t+1}) - H\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right), \tag{10}$$

where $H(\boldsymbol{Y}_{t+1}) = \mathbb{E}[-\log p(\boldsymbol{Y}_{t+1})]$ denotes the entropy and

$$H\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right) = \mathbb{E}_{\boldsymbol{Y}_{t+1}, \boldsymbol{Y}_t^{(p)}}\left[-\log p\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right)\right]$$

denotes the conditional entropy of $(\boldsymbol{Y}_{t+1})_t$ given its past.

If we assume $\boldsymbol{Y}_{t+1}$ to be normally distributed—which can be justified by the fact that it corresponds to a mixture of a potentially high number of distributions from the original high-dimensional space—then its differential entropy is given by $H(\boldsymbol{Y}_{t+1}) = \frac{1}{2}\log\{(2\pi e)^M\} + \log\{|\operatorname{cov}(\boldsymbol{Y}_{t+1})|\}$ and is thus a strictly increasing function of the determinant of its covariance. Now recall that $(\boldsymbol{X}_t)_t$ is assumed to have zero mean and covariance $\mathbf{I}$. Thus, $\operatorname{cov}(\boldsymbol{Y}_{t+1}) = \mathbf{I}$ holds independently of the selected transformation $\mathbf{A}$ which makes $H(\boldsymbol{Y}_{t+1})$ independent of $\mathbf{A}$ too.

What remains for the maximization of (10) is the minimization of the term $H(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)})$. Again assuming Gaussian distributions, the differential conditional entropy is given by

$$H\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right) = \frac{1}{2}\log\{(2\pi e)^M\} + \mathbb{E}_{\boldsymbol{Y}_t^{(p)}}\left[\log\left\{\left|\operatorname{cov}\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right)\right|\right\}\right]. \tag{11}$$

When we consider the special case of the conditional covariance $\text{cov}(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)} = \mathbf{y}_t^{(p)}) =: \Sigma_{\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}}$ being the same for every value that $\boldsymbol{Y}_t^{(p)}$ may take, then the expected value reduces to $\log\{|\Sigma_{\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}}|\}$ and (11) becomes minimal for the projection for which the resulting determinant $|\Sigma_{\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}}|$ is minimized. Furthermore, under this assumption, (1) can be written as

$$\begin{aligned}
&\text{trace}\left(\mathbb{E}_{\boldsymbol{Y}_t^{(p)}}\left[\text{cov}\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right)\right]\right)\\
&= \text{trace}\left(\Sigma_{\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}}\right)\\
&= \text{trace}\left(\mathbf{A}^T\Sigma_{\boldsymbol{X}_{t+1}|\boldsymbol{Y}_t^{(p)}}\mathbf{A}\right).
\end{aligned}$$

Thus, it becomes clear that $\Sigma_{\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}}$ with minimal trace is constructed by selecting the principle directions from the $N \times N$ matrix $\Sigma_{\boldsymbol{X}_{t+1}|\boldsymbol{Y}_t^{(p)}}$ that correspond to the M smallest eigenvalues. Thereby the determinant $|\Sigma_{\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}}|$ is minimized as well, since—like for the trace—its minimization only depends on the selection of the smallest eigenvalues. Thus, GPFA produces features with the maximum predictive information under this assumption of a prediction error $\Sigma_{\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}}$ independent of the value of $\boldsymbol{Y}_t^{(p)}$ (and to the degree that the iterated heuristic in Sect. 2.8 minimizes (1)).

For the general case of different $\text{cov}(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)})$ for different values of $\boldsymbol{Y}_t^{(p)}$ we have the following equality for the last term in (11):

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{Y}_t^{(p)}}\left[\log\left\{\left|\text{cov}\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right)\right|\right\}\right]\\
&= \mathbb{E}_{\boldsymbol{Y}_t^{(p)}}\left[\text{trace}\left(\log\left\{\text{cov}\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right)\right\}\right)\right]\\
&= \text{trace}\left(\mathbb{E}_{\boldsymbol{Y}_t^{(p)}}\left[\log\left\{\text{cov}\left(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t^{(p)}\right)\right\}\right]\right). \qquad (12)
\end{aligned}$$

This corresponds to GPFA's objective (1) with logarithmically weighted covariances. Such a weighting intuitively makes sense from the perspective that the predictive information expresses how many bits of uncertainty (that is, variance) are removed through knowing about the feature's past. Since the number of bits only grows logarithmically with increasing variance, the weight of events with low uncertainty is disproportionally large, which could be accounted for in the objective function if the goal would be low coding length instead of low future variance.

2.10 Time complexity

In the following section we derive GPFA's asymptotic time complexity in dependence of the number of training samples S, input dimensions N, process order p, output dimensions M, number of iterations R, as well as the neighborhood size k.

2.10.1 k-nearest-neighbor search

The first computationally expensive step of the GPFA is the k-nearest-neighbor search. When we naively assume a brute-force approach, it can be realized in $\mathcal{O}(NpS)$. This search is repeated for each of the S data points and for each of the R iterations (in $N \cdot p$ dimensions for the first iteration and in $M \cdot p$ for all others). Thus, the k-nearest-neighbor search in the worst case has a time complexity of

$$\mathcal{O}(NpS^2 + RMpS^2).$$

Of course, more efficient approaches to k-nearest-neighbor search exist.

2.10.2 Matrix multiplications

The second expensive step consists of the matrix multiplications in (8) to calculate the projected graph Laplacians. For a multiplication of two dense matrices of size $l \times m$ and $m \times n$ we assume a computational cost of $\mathcal{O}(lmn)$. If the first matrix is sparse, with L being the number of non-zero elements, we assume $\mathcal{O}(Ln)$. This gives us a complexity of $\mathcal{O}(N^2S + LN)$ for the left-hand side of (8). For GPFA (1) there is a maximum of $L = 2k^2S$ non-zero elements (corresponding to the edges added to the graph, which are not all unique), for GPFA (2) there is a maximum of $L = 2kS$. The right-hand side of (8) then can be ignored since it's complexity of $\mathcal{O}(N^2S + SN)$ is completely dominated by the left-hand side. Factoring in the number of iterations R, we finally have computational costs of

$$\mathcal{O}(RN^2S + RLN)$$

with $L = k^2S$ for GPFA (1) and $L = kS$ for GPFA (2).

2.10.3 Eigenvalue decomposition

For solving the eigenvalue problem (8) R times we assume an additional time complexity of $\mathcal{O}(RN^3)$. This is again a conservative guess because only the first M eigenvectors need to be calculated.

2.10.4 Overall time complexity

Taking together the components above, GPFA has a time complexity of $\mathcal{O}(NpS^2 + RMpS^2 + RN^2S + RLN + RN^3)$ with $L = k^2S$ for GPFA (1) and $L = kS$ for GPFA (2). In terms of the individual variables, that is: $\mathcal{O}(S^2)$, $\mathcal{O}(N^3)$, $\mathcal{O}(M)$, $\mathcal{O}(p)$, $\mathcal{O}(R)$, and $\mathcal{O}(k^2)$ or $\mathcal{O}(k)$ for GPFA (1) or GPFA (2), respectively.

3 Related methods

In this section we briefly summarize the algorithms most closely related to GPFA, namely SFA, ForeCA, and PFA.

3.1 SFA

Although SFA originally has been developed to model aspects of the visual cortex, it has been successfully applied to different problems in technical domains as well (see Escalante et al. 2012 for a short overview), like, for example, state-of-the art age-estimation (Escalante et al. 2016). It is one of the few DR algorithms that considers the temporal structure of the data. In particular, slowly varying signals can be seen as a special case of predictable features (Creutzig and Sprekeler 2008). It is also possible to reformulate the slowness principle implemented by SFA in terms of graph embedding, for instance to incorporate label information into the optimization problem (Escalante et al. 2013).

Adopting the notation from above, SFA finds an orthogonal transformation $\mathbf{A} \in \mathbb{R}^{N\times M}$ such that the extracted signals $\mathbf{y}_t = \mathbf{A}^T\mathbf{x}_t$ have minimum temporal variation $\langle||\mathbf{y}_{t+1} - \mathbf{y}_t||^2\rangle_t$. The input vectors $\mathbf{x}_t$—and thus $\mathbf{y}_t$ as well—are assumed to be white.

3.2 ForeCA

In case of ForeCA (Goerg 2013), $(\boldsymbol{X}_t)_t$ is assumed to be a stationary second-order process and the goal of the algorithm is finding an extraction vector $\mathbf{a}$ such that the projected signals $Y_t = \mathbf{a}^T\boldsymbol{X}_t$ are as *forecastable* as possible, that is, having a low entropy in their power spectrum. Like SFA, ForeCA has the advantage of being completely model- and parameter-free.

For the formal definition of *forecastability*, first consider the signal's autocovariance function $\gamma_Y(l) = E(Y_t - \mu_Y)E(Y_{t-l} - \mu_Y)$, with μ_Y being the mean value and the corresponding autocorrelation function $\rho_Y(l) = \gamma_Y(l)/\gamma_Y(0)$. The spectral density of the process can be calculated as the Fourier transform of the autocorrelation function, i.e., as

$$f_Y(\lambda) = \sum_{j=-\infty}^{\infty} \rho_Y(j)e^{ij\lambda},$$

with $i = \sqrt{-1}$ being the imaginary unit.

Since $f_Y(\lambda) \geq 0$ and $\int_{-\pi}^{\pi} f_Y(\lambda)d\lambda = 1$, the spectral density can be interpreted as a probability density function and thus its entropy calculated as

$$H(Y_t) = -\int_{-\pi}^{\pi} f_Y(\lambda)\log(f_Y(\lambda))d\lambda.$$

For white noise the spectral density becomes uniform with entropy $\log(2\pi)$. This motivates the definition of *forecastability* as

$$\Omega(Y_t) := 1 - \frac{H(Y_t)}{\log(2\pi)},$$

with values between 0 (white noise) and ∞ (most predictable). Since $\Omega(Y_t) = \Omega(\mathbf{a}^T\boldsymbol{X}_t)$ is invariant to scaling and shifting, $\boldsymbol{X}_t$ can be assumed to be white, without loss of generality. The resulting optimization problem

$$\arg\max_{\mathbf{a}} \Omega\left(\mathbf{a}^T\boldsymbol{X}_t\right)$$

then is solved by an EM-like algorithm that uses weighted overlapping segment averaging (WOSA) to estimate the spectral density of a given (training) time series. By subsequently finding projections which are orthogonal to the already extracted ones, the approach can be employed for finding projections to higher dimensional subspaces as well. For details about ForeCA (see Goerg 2013).

3.3 PFA

The motivation behind PFA is finding an orthogonal transformation $\mathbf{A} \in \mathbb{R}^{N\times M}$ as well as coefficient matrices $\mathbf{B}_i \in \mathbb{R}^{M\times M}$, with $i = 1\ldots p$, such that the linear, autoregressive prediction error of order p,

$$\left\langle \|\mathbf{A}^T\mathbf{x}_t - \sum_{i=1}^{p} \mathbf{B}_i\mathbf{A}^T\mathbf{x}_{t-i}\|^2 \right\rangle_t,$$

is minimized. However, this is a difficult problem to optimize because the optimal values of $\mathbf{A}$ and $\mathbf{B}_i$ mutually depend on each other. Therefore the solution is approached via a related but easier optimization problem: Let $\zeta_t := (\mathbf{x}_{t-1}^T, \ldots, \mathbf{x}_{t-p}^T)^T \in \mathbb{R}^{N \cdot p}$ be a vector containing the p-step history of $\mathbf{x}_t$. Let further $\mathbf{W} \in \mathbb{R}^{N \times N \cdot p}$ contain the coefficients that minimize the error of predicting $\mathbf{x}_t$ from its own history, i.e., $\langle \|\mathbf{x}_t - \mathbf{W}\zeta_t\|^2 \rangle_t$. Then minimizing $\langle \|\mathbf{A}^T\mathbf{x}_t - \mathbf{A}^T\mathbf{W}\zeta_t\|^2 \rangle_t$ with respect to $\mathbf{A}$ corresponds to a PCA (in the sense of finding the directions of smallest variance) on that prediction error. Minimizing this prediction error however does not necessarily lead to features $\mathbf{y}_t = \mathbf{A}^T\mathbf{x}_t$ that are best for predicting their own future because the calculated prediction was based on the history of $\mathbf{x}_t$, not $\mathbf{y}_t$ alone. Therefore an additional heuristic is proposed that is based on the intuition that the inherited errors of K times repeated autoregressive predictions create an even stronger incentive to avoid unpredictable components. Finally,

$$\sum_{i=0}^{K} \left\langle \|\mathbf{A}^T\mathbf{x}_t - \mathbf{A}^T\mathbf{W}\mathbf{V}^i\zeta_t\|^2 \right\rangle_t$$

is minimized with respect to $\mathbf{A}$, where $\mathbf{V} \in \mathbb{R}^{N \cdot p \times N \cdot p}$ contains the coefficients that minimize the prediction error $\langle \|\zeta_{t+1} - \mathbf{V}\zeta_t\|^2 \rangle_t$.

Like the other algorithms, PFA includes a preprocessing step to whiten the data. So far, PFA has been shown to work on artificially generated data. For further details about PFA (see Richthofer and Wiskott 2013).

4 Experiments

We conducted experiments[4] on different datasets to compare GPFA to SFA, ForeCA, and PFA. As a baseline, we compared the features extracted by all algorithms to features that were created by projecting into an arbitrary (i.e., randomly selected) M-dimensional subspace of the data's N-dimensional vector space.

For all experiments, first the training set was whitened and then the same whitening transformation was applied to the test set. After training, the learned projection was used to extract the most predictable M-dimensional signal from the test set with each of the algorithms. The extracted signals were evaluated in terms of their empirical predictability (2). The neighborhood size used for this evaluation is called q in the following to distinguish it from the neighborhood size k used during the training of GPFA. Since there is no natural choice for the different evaluation functions that effectively result from different q, we arbitrarily chose $q = 10$ but also include plots on how results change with the value of q. The size of training and test set will be denoted by S_{train} and S_{test}, respectively. The plots show mean and standard deviation for 50 repetitions of each experiment.[5]

4.1 Toy example ("predictable noise")

We created a small toy data set to demonstrate performance differences of the different algorithms. The data set contains a particular kind of predictable signals which are challenging to identify for most algorithms. Furthermore, the example is suited to get an impression for

[4] GPFA and experiments have been implemented in Python 2.7. Code and datasets will be published upon acceptance.

[5] Note that while the algorithms themselves do not depend on any random effects, the data set generation does.

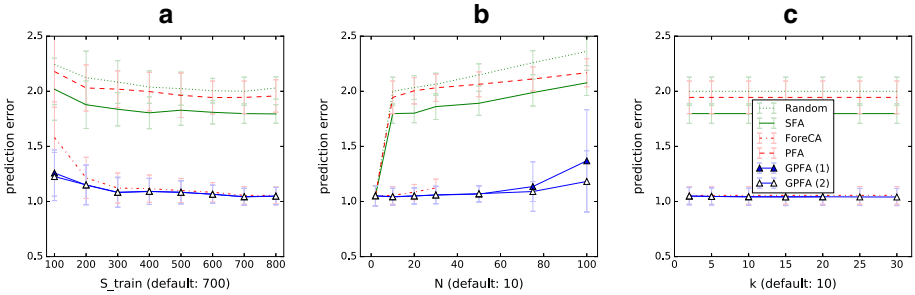

Fig. 3 Predictability in terms of (2) of two dimensional signals ($M = 2$) extracted from the the toy dataset by the different algorithms. If not varied during the experiment, parameters were $p = 1$, $k = 10$, $q = 10$, $S_{train} = 700$, $S_{test} = 100$, $N = 10$, $R = 50$, and $K = 0$

running time constants of the different algorithms that are not apparent from the big $\mathcal{O}$ notation in Sect. 2.10.

First, a two-dimensional signal

$$\mathbf{x}_t = \begin{pmatrix} \xi_t \\ \xi_{t-1} \end{pmatrix} \tag{13}$$

was generated with ξ_t being normally distributed noise. Half of the variance in this sequence can be predicted when $\mathbf{x}_{t-1}$ is known (i.e., $p = 1$), making the noise partly predictable. This two-dimensional signal was augmented with $N - 2$ additional dimensions of normally distributed noise to create the full data set. We generated such data sets with up to $S_{train} = 800$ training samples, a fixed test set size of $S_{test} = 100$, and with up to $N = 100$ input dimensions and extracted $M = 2$ components with each of the algorithms. If not varied themselves during the experiment, values were fixed to $S_{train} = 700$ training samples, $N = 10$ input dimensions, and $k = 10$ neighbors for the training of GPFA. The results of PFA did not change significantly with number of iterations K, which was therefore set to $K = 0$.

Figure 3 shows the predictability of the signals extracted by the different algorithms and how it varies in S_{train}, N, and k. Only ForeCA and GPFA are able to distinguish the two components of predictable noise from the unpredictable ones, as can be seen from reaching a variance of about 1, which corresponds to the variance of the two generated, partly predictable components. As Fig. 3b shows, the performance of both versions of GPFA (as of all other algorithms) declines with a higher number of input dimensions (but for GPFA (2) less than for GPFA (1)). At this point, a larger number of training samples is necessary to produce more reliable results (experiments not shown). The results do not differ much with the choice of k though.

As the runtime plots of the experiments reveal (see Fig. 4), ForeCA scales especially badly in the number of input dimensions N, so that it becomes very computationally expensive to be applied to time series with more than a few dozen dimensions. For that reason we excluded ForeCA from the remaining, high-dimensional experiments.

4.2 Auditory data

In the second set of experiments we focused on short-time Fourier transforms (STFTs) of audio files. Three public domain audio files (a silent film piano soundtrack, ambient sounds from a bar, and ambient sounds from a forest) were re-sampled to 22kHz mono. The STFTs were calculated with the Python library `stft` with a frame length of 512 and a cosine window

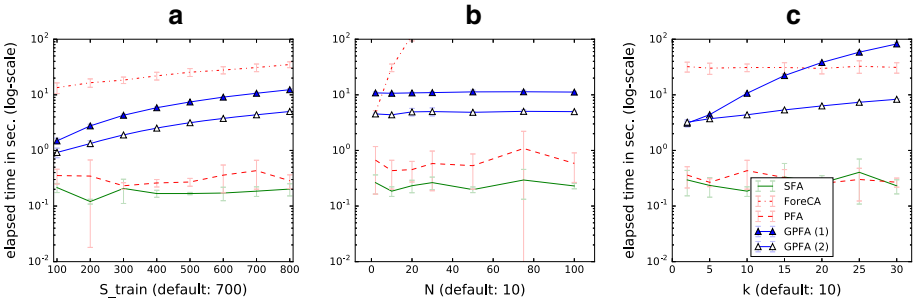

Fig. 4 Runtime for the experiments in Fig. 3

function, resulting in three datasets with 26,147, 27,427, and 70,433 frames, respectively, each with 512 dimensions (after discarding complex-conjugates and representing the remaining complex values as two real values each). For each repetition of the experiment, $S_{train} = 10{,}000$ successive frames were picked randomly as training set and $S_{test} = 5000$ distinct and successive frames were picked as test set. PCA was calculated for each training set to preserve 99% of the variance and this transformation was applied to training and test set alike.

The critical parameters p and k, defining the assumed order of the process and the neighborhood size respectively, were selected through cross-validation to be a good compromise between working well for all values of M and also not treating one of the algorithms unfavourably. PFA and GPFA tend to benefit from the same values for p. The number of iteration R for GPFA was found to be not very critical and was set to $R = 50$. The iteration parameter K of PFA was selected by searching for the best result in $\{0 \ldots 10\}$, leaving all other parameters fixed.

The central results can be seen in Figs. 5f, 6f, 7f in terms of the predictability of the components extracted by the different algorithms in dependence of their dimensionality M. The other plots show how the results change with the individual parameters. Increasing the number of past time steps p tends to improve the results first but may let them decrease later (see Figs. 5a, 6a, 7a). Presumably, because higher numbers of p make the models more prone to overfitting. The neighborhood size k had to be selected carefully for each of the different datasets. While its choice was not critical on the first dataset, the second dataset benefited from low values for k and the third one from higher values (see Figs. 5b, 6b, 7b). Similar, the neighborhood size q for calculating the final predictability of the results had different effects for different datasets (see Figs. 5c, 6c, 7c). At this point it's difficult to favor one value over another, which is why we kept q fixed to $q = 10$. As expected, results tend to improve with increasing numbers of training samples S_{train} (see Figs. 5d, 6d, 7d). Similarly, results first improve with the number of iterations R for GPFA and then remain stable (see Figs. 5e, 6e, 7e). We take this as evidence for the viability of the iteration heuristic motivated in Sect. 2.8.

To gauge the statistical reliability of the results, we applied the Wilcoxon signed-rank test, testing the null hypothesis that the results for different pairs of algorithms actually come from the same distribution. We tested this hypothesis for each data set for the experiment with default parameters, i.e., for the results shown in Figs. 5f, 6f, 7f with $M = 5$. As can be seen from the p-values in Table 1, the null hypothesis can be rejected with certainty in many cases, which confirms that GPFA (2) learned the most predictable features on two of three datasets. For GPFA (1) the results are clear for the first dataset as well for the second in comparison to PFA. It remains a small probability, however, that the advantage compared to SFA on the second dataset is only due to chance. For the large third dataset, all algorithms

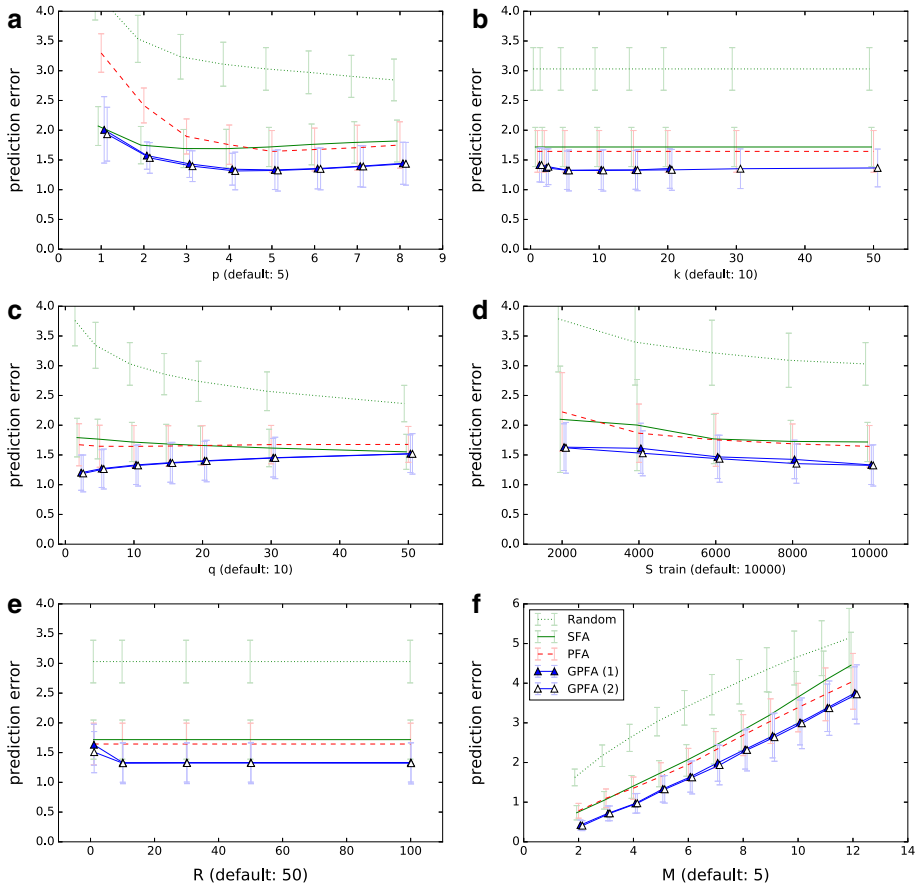

Fig. 5 Results for STFT #1 (*"piano"*): if not varied during the experiment, parameters were $p = 5$, $k = 10$, $q = 10$, $S_{train} = 10,000$, $R = 50$, $M = 5$, and $K = 10$. Slight x-shifts have been induced to separate *error bars*

produce relatively similar results with high variance between experiments. It depends on the exact value of M if SFA or GPFA produced the best results. For $M = 5$ GPFA happened to find slightly more predictable results (not highly significant though as can be seen in Table 1). But in general we don't see a clear advantage of GPFA on the third dataset.

4.3 Visual data

A third experiment was conducted on a visual dataset. We modified the simulator from the *Mario AI challenge* (Karakovskiy and Togelius 2012) to return raw visual input in gray-scale without text labels. The raw input was scaled from 320×240 down to 160×120 dimensions and then the final data points were taken from a small window of $20 \times 20 = 400$ pixels at a position where much of the game dynamics happened (see Fig. 8 for an example). As with the auditory datasets, for each experiment $S_{train} = 10{,}000$ successive training and $S_{test} = 5000$ non-overlapping test frames were selected randomly and PCA was applied to both, preserving 99% of the variance. Eventually, M predictable components were extracted by each of the

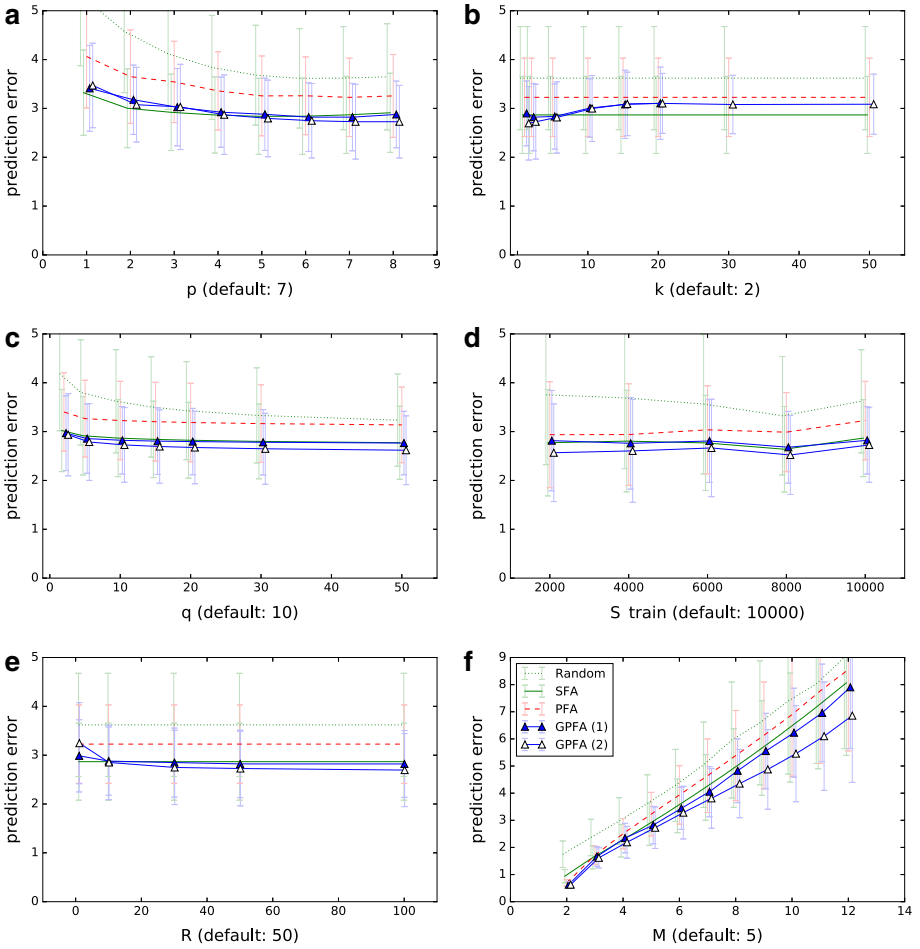

Fig. 6 Results for STFT #2 (*"bar"*): if not varied during the experiment, parameters were $p = 7$, $k = 2$, $q = 10$, $S_{train} = 10,000$, $R = 50$, $M = 5$, and $K = 10$. Slight x-shifts have been induced to separate *error bars*

algorithms and evaluated with respect to their predictability (2). Parameters p and k again were selected from a range of candidate values to yield the best results (see Fig. 9a, b).

Two things are apparent from the results as shown in Fig. 9. First, the choice of parameters was less critical compared to the auditory datasets. And second, all compared algorithms show quite similar results in terms of their predictability. GPFA only is able to find features slightly more predictable than those of SFA for higher values of M (see Fig. 9f). Again, this observation is highly significant with a Wilcoxon p-value of 0.00 for $M = 12$.

5 Discussion and future work

In the previous section we saw that GPFA produced the most predictable features on a toy example with a certain kind of predictable noise as well as on two auditory datasets.

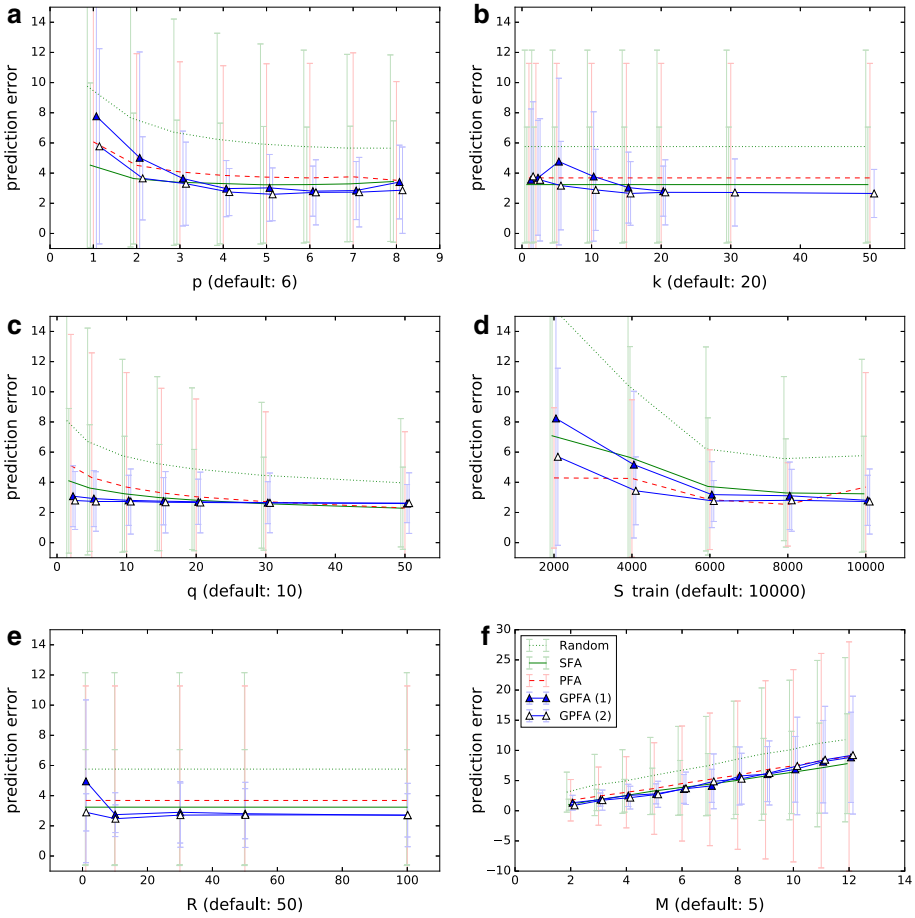

Fig. 7 Results for STFT #3 (*"forest"*): if not varied during the experiment, parameters were $p = 6$, $k = 20$, $q = 10$, $S_{train} = 10,000$, $R = 50$, $M = 5$, and $K = 10$. Slight x-shifts have been induced to separate *error bars*

Table 1 p-values for the Wilcoxon signed-rank test which tests the null hypothesis that a pair of samples come from the same distribution

	STFT #1		STFT #2		STFT #3	
	SFA	PFA	SFA	PFA	SFA	PFA
GPFA (1)	**0.00**	**0.00**	0.18	**0.00**	0.43	0.09
GPFA (2)	**0.00**	**0.00**	**0.00**	**0.00**	0.38	0.17

Values refer to the experiments shown in Figs. 5f, 6f, 7f with $M = 5$. Row and column indicate the pair of algorithms compared. p-values that show a significant ($p \leq 0.01$) advantage of GPFA over the compared algorithm are printed bold

However, on a third auditory dataset as well as on a visual dataset, GPFA did not show a clear advantage compared to SFA. This matches our experience with other visual datasets (not shown here). We hypothesize that SFA's assumption of the most relevant signals being

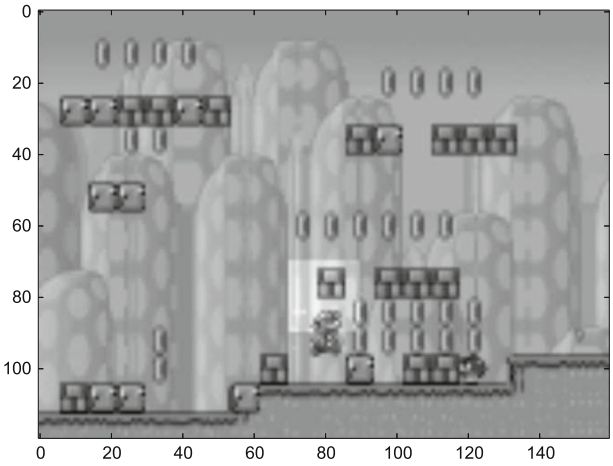

Fig. 8 An example frame generated by a modified simulator from the Mario AI challenge. The *highlighted square* indicates the 400 pixels extracted for the experiment

the slow ones may especially suited for the characteristics of visual data. This also matches the fact that SFA originally was designed for and already proved to work well for signal extraction from visual data sets. A detailed analysis of which algorithm and corresponding measure of predictability is best suited for what kind of data or domain remains a subject of future research.

In practical terms we conclude that GPFA (2) has some advantages over GPFA (1). First, its linear time complexity in k (see Sect. 2.10.2) makes a notable difference in practice (see Sect. 4.1). Second, GPFA (2) consistently produced better results (see Sect. 4) which is a bit surprising given that the fully connected graph of GPFA (1) is theoretically more sound and also matches the actual evaluation criterion (2). Our intuition here is that it is beneficial to give $\mathbf{y}_{t+1}$ a central role in the graph because it is a more reliable estimate of the true mean of $p(\boldsymbol{Y}_{t+1}|\boldsymbol{Y}_t = \mathbf{y}_t)$ than the empirical mean of all data points (stemming from different distributions) in the fully connected graph.

In the form described above, GPFA performs linear feature extraction. However, we are going to point out three strategies to extend the current algorithm for non-linear feature extraction. The first strategy is very straight-forward and can be applied to the other linear feature extractors as well: in a preprocessing step, the data is expanded in a non-linear way, for instance through all polynomials up to a certain order. Afterwards, application of a linear feature extractor implicitly results in non-linear feature extraction. This strategy is usually applied to SFA, often in combination with hierarchical stacking of SFA nodes which further increases the non-linearities while at the same time regularizing spatially (on visual data) (Escalante et al. 2012).

The other two approaches to non-linear feature extraction build upon the graph embedding framework. We already mentioned above that kernel versions of graph embedding are readily available (Yan et al. 2007; Cai et al. 2007). Another approach to non-linear graph embedding was described for an algorithm called *hierarchical generalized SFA*: A given graph is embedded by first expanding the data in a non-linear way and then calculating a lower-dimensional embedding of the graph on the expanded data. This step is repeated—each time with the original graph—resulting in an embedding for the original graph that is increasingly non-linear with every repetition (see Sprekeler 2011 for details).

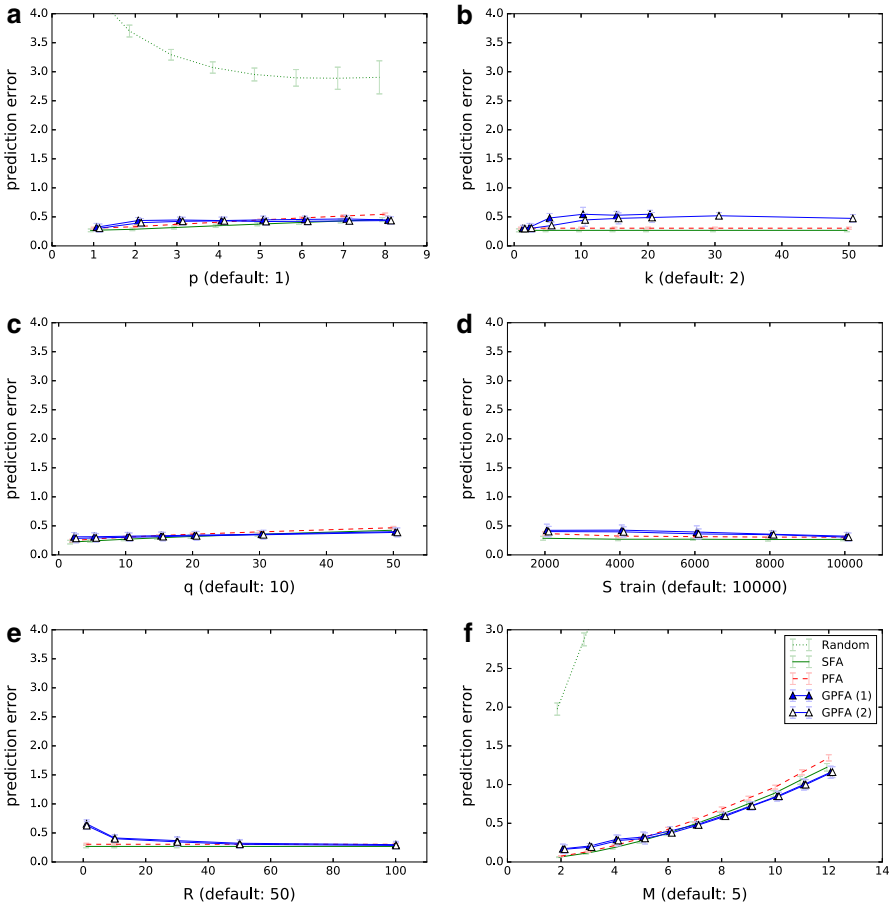

Fig. 9 Results for visual dataset (*"Super Mario"*): if not varied during the experiment, parameters were $p = 1, k = 2, q = 10, S_{train} = 10,000, R = 50, M = 5$, and $K = 1$. Slight x-shifts have been induced to separate *error bars*

Regarding the analytical understanding of GPFA, we have shown in Sect. 2.9 under which assumptions GPFA can be understood as finding the features with the highest predictive information, for instance when the underlying process is assumed to be deterministic but its states disturbed by independent Gaussian noise. If we generally had the goal of minimizing the coding length of the extracted signals (which would correspond to high predictive information) rather than minimizing their next-step variance, then the covariances in GPFA's main objective (1) needed to be weighted logarithmically. Such an adoption, however, would not be straight forward to include into the graph structure.

Another information-theoretic concept relevant in this context (besides predictive information) is that of information bottlenecks (Tishby et al. 2000). Given two random variables $\boldsymbol{A}$ and $\boldsymbol{B}$, an information bottleneck is a compressed variable $\boldsymbol{T}$ that solves the problem $\min_{p(\mathbf{t}|\mathbf{a})} I(\boldsymbol{A}; \boldsymbol{T}) - \beta I(\boldsymbol{T}; \boldsymbol{B})$. Intuitively, $\boldsymbol{T}$ encodes as much information from $\boldsymbol{A}$ about $\boldsymbol{B}$ as possible while being restricted in complexity. When this idea is applied to time series such that $\boldsymbol{A}$ represents the past and $\boldsymbol{B}$ the future, then $\boldsymbol{T}$ can be understood as encoding the most predictable aspects of that time series. In fact, SFA has been shown to implement a spe-

cial case of such a past-future information bottleneck for Gaussian variables (Creutzig and Sprekeler 2008). The relationship between GPFA and (past-future) information bottlenecks shall be investigated in the future.

In Sect. 2.6 we introduced the heuristic of reducing the variance of the past in addition that of the future. Effectively this groups together parts of the feature space that have similar expected futures. This property may be especially valuable for interactive settings like reinforcement learning. When you consider an agent navigating its environment, it is usually less relevant to know which way it reached a certain state but rather where it can go to from there. That's why state representations encoding the agent's future generalize better and allow for more efficient learning of policies than state representations that encode the agent's past (Littman et al. 2001; Rafols et al. 2005). To better address interactive settings, multiple actions may incorporated into GPFA by conditioning the kNN search on actions, for instance. Additional edges in the graph could also allow grouping together features with similar expected rewards. We see such extension of GPFA as an interesting avenue of future research.

6 Conclusion

We presented *graph-based predictable feature analysis* (GPFA), a new algorithm for unsupervised learning of predictable features from high-dimensional time series. We proposed to use the variance of the conditional distribution of the next time point given the previous ones to quantify the predictability of the learned representations and showed how this quantity relates to the information-theoretic measure of predictive information. As demonstrated, searching for the projection that minimizes the proposed predictability measure can be reformulated as a problem of graph embedding. Experimentally, GPFA produced very competitive results, especially on auditory STFT datasets, which makes it a promising candidate for every problem of dimensionality reduction (DR) in which the data is inherently embedded in time.

References

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396.

Bialek, W., Nemenman, I., & Tishby, N. (2001). Predictability, complexity, and learning. *Neural Computation*, *13*(11), 2409–2463.

Bialek, W., & Tishby, N. (1999). *Predictive information*. e-print arXiv:cond-mat/9902341, February 1999.

Cai, D., He, X., & Han, J. (2007). Spectral regression: A unified approach for sparse subspace learning. In 7th IEEE *International Conference on Data Mining* (ICDM 2007), pp. 73–82. IEEE.

Collomb, G. (1985). Non parametric time series analysis and prediction: Uniform almost sure convergence of the window and k-nn autoregression estimates. *Statistics: A Journal of Theoretical and Applied Statistics*, *16*(2), 297–307.

Creutzig, F., & Sprekeler, H. (2008). Predictive coding and the slowness principle: An information-theoretic approach. *Neural Computation*, *20*(4), 1026–1041.

Escalante, B., Alberto, N., & Wiskott, L. (2012). Slow feature analysis: Perspectives for technical applications of a versatile learning algorithm. *Künstliche Intelligenz (Artificial Intelligence)*, *26*(4), 341–348.

Escalante, B., Alberto, N., & Wiskott, L. (2013). How to solve classification and regression problems on high-dimensional data with a supervised extension of slow feature analysis. *Journal of Machine Learning Research*, *14*(1), 3683–3719.

Escalante, B., Alberto, N., & Wiskott, L. (2016) *Improved graph-based SFA: Information preservation complements the slowness principle*. e-print arXiv:1601.03945, January 2016.

Goerg, G. (2013). Forecastable component analysis. In *Proceedings of the 30th international conference on machine learning (ICML 2013)*, (Vol 28, pp. 64–72). JMLR Workshop and Conference Proceedings.

Han, F., & Liu, H. (2013). Principal component analysis on non-gaussian dependent data. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, (Vol. 28, pp. 240–248). JMLR Workshop and Conference Proceedings.

He, X., & Niyogi, P. (2004). Locality preserving projections. In T. Sebastian, K. S. Lawrence, & S. Bernhard (Eds.), *Advances in neural information processing systems* (Vol. 16, pp. 153–160). Cambridge, MA: MIT Press.

Jonschkowski, R., & Brock, O. (2015). Learning state representations with robotic priors. *Autonomous Robots*, *39*(3), 407–428.

Karakovskiy, S., & Togelius, J. (2012). The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 55–67.

Littman, M. L., Sutton, R. S., & Singh, S. (2001). Predictive representations of state. In *Advances in neural information processing systems (NIPS)* (Vol. 14, pp. 1555–1561). Cambridge, MA: MIT Press.

Rafols, E. J., Ring, M. B., Sutton, R. S., & Tanner, B. (2005). Using predictive representations to improve generalization in reinforcement learning. In *Proceedings of the 19th international joint conference on Artificial intelligence, IJCAI'05* (pp. 835–840). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Richthofer, S., & Wiskott, L. (2013). *Predictable feature analysis*. e-print arXiv:1311.2503, November 2013.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Shalizi, C. R., & Crutchfield, J. P. (2001). Computational mechanics: Pattern and prediction, structure and simplicity. *Journal of Satistical Physics*, *104*(3–4), 817–879.

Sprague, N. (2009). Predictive projections. In *Proceedings of the 21st international joint conference on artifical intelligence (IJCAI 2009)* (pp. 1223–1229). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Sprekeler, H. (2011). On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, *23*(12), 3287–3302.

Still, S. (2009). Information-theoretic approach to interactive learning. *Europhysics Letters*, *85*(2), 28005.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Tishby, N., Pereira, F. C., & Bialek, W. (2000). *The information bottleneck method*. e-print arXiv:physics/0004057, April 2000.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416.

Wiskott, L., & Sejnowski, T. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, *14*(4), 715–770.

Yan, S., Dong, X., Zhang, B., Zhang, H.-J., Yang, Q., & Lin, S. (2007). Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(1), 40–51.