Functional Autoencoder

Peida Wu

IMS

December 30, 2025

Functional Data & Motivation

$$X(t) \approx \sum_{k=1}^{n} c_k \phi_k(t)$$

where $\phi_k(t)$ is basis functions.(e.g. Fourier basis, B-spline basis)
Linear method for representation learning like **FPCA** suffices.
In reality, each dimension of functional data is nonlinear and the correlation is nonlinear.

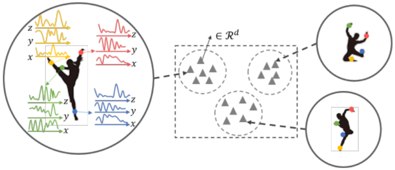


Figure 1: example of functional data

Functional representation Learning

Some notations:

$$\mathbf{x} = [x_1(t), x_2(t), \ldots, x_P(t)] \in \mathcal{H}^P$$

where $\mathcal{H}$ denotes the function space, a subspace of L^2.$x_j \in \mathcal{H}^P$ is a function.

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$$

is n samples of functional data.

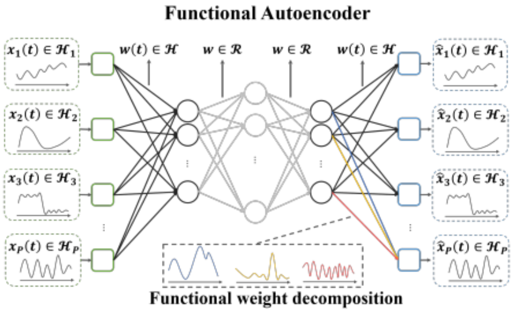
$$\mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_d] \in \mathbb{R}^d$$

denotes d dimension of encoding.

Functional Autoencoder

$\phi : \mathcal{H}^P \to \mathbb{R}^d$ is the encoder where $\mathbb{R}^d$ is a vector space.
$\psi : \mathbb{R}^d \to \mathcal{H}^P$ is the decoder.

$$\phi, \psi = \arg\min_{\phi,\psi} \|\mathbf{x} - (\psi \circ \phi)\mathbf{x}\|^2.$$



Functional Autoencoder
$x_1(t) \in \mathcal{H}_1$
$x_2(t) \in \mathcal{H}_2$
$x_3(t) \in \mathcal{H}_3$
$x_P(t) \in \mathcal{H}_P$
$w(t) \in \mathcal{H}$
$w \in \mathcal{R}$
$w \in \mathcal{R}$
$w(t) \in \mathcal{H}$
$\hat{x}_1(t) \in \mathcal{H}_1$
$\hat{x}_2(t) \in \mathcal{H}_2$
$\hat{x}_3(t) \in \mathcal{H}_3$
$\hat{x}_P(t) \in \mathcal{H}_P$
Functional weight decomposition

Figure 2: Functional Autoencoder

Functional Autoencoder

Encoder

$$O_k^{(1)} = \sigma\left(\sum_{j=1}^{P} \left\langle x_j(t), w_{k,j}^{(1)}(t) \right\rangle\right) = \sigma\left(\sum_{j=1}^{P} \int_{\tau_j} x_j(t)\, w_{k,j}^{(1)}(t)\, dt\right)$$

Decoder

$$\hat{x}_j(t) = \sum_k O_k^{(l-1)}\, w_{j,k}^{(l-1)}(t), \qquad j = 1, \ldots, P.$$

- $x_j(t)$input function$w_{k,j}^{(1)}(t), w_{j,k}^{(l-1)}(t)$functional weight
- σactivation function$O_k^{(1)}, O_k^{(l-1)}$: First and last hidden layer.

Loss function & distribution

$$\begin{aligned}\mathcal{L}(\Omega) &= \frac{1}{2n}\sum_{i=1}^{n}\left\|x_i - \hat{x}_i\right\|_2^2 \\ &= \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{P}\left\|x_{i,j}(t) - \hat{x}_{i,j}(t)\right\|_2^2 \\ &= \frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{P}\int_{\tau_j}\left(x_{i,j}(t) - \hat{x}_{i,j}(t)\right)^2 dt.\end{aligned} \tag{1}$$

Distribution:

1. FAE generalize the Autoencoder
2. is equivalent to FPCA when the activation function is linear.

Discrete Encoder: Numerical Approximation

- Replace the integral in the first layer with a numerical sum.
- For a common grid $\{t_1, \ldots, t_J\}$ with weights ω_j:

$$h_k^{(1)} = g\left(\int_{\mathcal{T}} X(t)\, w_k^{(l)}(t)\, dt\right) \approx g\left(\sum_{j=1}^{J} \omega_j X(t_j)\, w_k^{(l)}(t_j)\right).$$

- The input layer is now the vector

$$\big(X(t_1), \ldots, X(t_J)\big) \in \mathbb{R}^J.$$

- We still need a convenient way to represent and learn the weight functions $w_k^{(l)}(t)$.

Basis Expansion of Weight Functions

- Represent each functional weight by a basis expansion:

$$w_k^{(l)}(t) = \sum_{m=1}^{M^{(l)}} c_{mk}^{(l)}\, \phi_m^{(l)}(t),$$

  where $\{\phi_m^{(l)}(t)\}$ is a fixed set of basis functions (e.g., B-splines).
- On the discrete grid:

$$w_k^{(l)}(t_j) = \sum_{m=1}^{M^{(l)}} c_{mk}^{(l)}\, \phi_m^{(l)}(t_j).$$

- The trainable parameters are the coefficients $c_{mk}^{(l)}$ instead of the whole functions $w_k^{(l)}(\cdot)$.
- Plugging this into the discrete encoder leads naturally to a new intermediate layer: the *feature layer*.

Feature Layer in the Encoder

- $$h_k^{(1)} = g\Big(\sum_{j=1}^{J} \omega_j X(t_j) \sum_{m=1}^{M^{(l)}} c_{mk}^{(l)}\, \phi_m^{(l)}(t_j)\Big).$$
- Exchange the sums:
$$h_k^{(1)} = g\Big(\sum_{m=1}^{M^{(l)}} c_{mk}^{(l)} \sum_{j=1}^{J} \omega_j X(t_j)\, \phi_m^{(l)}(t_j)\Big).$$
- Define the **feature layer** entries
$$h_k^{(1)} = g\left(\sum_{m=1}^{M^{(l)}} c_{mk}^{(l)}\, f_m\right).$$
where
$$f_m = \sum_{j=1}^{J} \omega_j X(t_j)\, \phi_m^{(l)}(t_j), \quad m = 1, \ldots, M^{(l)}.$$

Handling Irregular Observation Grids

- Different functions X_i may be observed at different time points $\{t_{ij}\}_{j=1}^{J_i}$ (irregular grids).
- The feature layer generalizes to

$$f_{im} = \sum_{j=1}^{J_i} \omega_{ij} \, X_i(t_{ij}) \, \phi_m^{(I)}(t_{ij}), \quad m = 1, \ldots, M^{(I)}.$$

- For each curve, we still obtain a feature vector $f_i \in \mathbb{R}^{M^{(I)}}$ of the same dimension.
- The rest of the network (hidden layers) does not need to care about the individual time grids:
  - irregular sampling is handled entirely by the feature layer,
  - downstream layers see a standard fixed-length input vector.

Decoder and Coefficient Layer

- Expand the output weight functions with another basis:

$$w_k^{(O)}(t) = \sum_{m=1}^{M^{(O)}} c_{mk}^{(O)}\, \phi_m^{(O)}(t).$$

- Discrete reconstruction at t_j:

$$\hat{X}(t_j) = \sum_{k=1}^{K^{(L)}} h_k^{(L)} w_k^{(O)}(t_j) = \sum_{m=1}^{M^{(O)}} b_m\, \phi_m^{(O)}(t_j),$$

where

$$b_m = \sum_{k=1}^{K^{(L)}} h_k^{(L)}\, c_{mk}^{(O)}.$$

- The vector $b = (b_1, \dots, b_{M^{(O)}})$ defines the **coefficient layer**, which parameterizes the reconstructed function in the output basis.

Advantages of the Discrete FAE Construction

- The feature and coefficient layers embed functional structure:
  - inputs are processed through numerical integration and input basis functions,
  - outputs are reconstructed via smooth basis functions.
- The model:
  - works directly on noisy discrete samples,
  - naturally handles irregular time grids,
  - produces smooth functional reconstructions,
  - and yields low-dimensional representations in the hidden layers.

Connection to FPCA

- Consider a **linear** FAE with one hidden layer.
- Encoder:

$$z_{ik} = \int_T X_i(t)\, w_k^{(I)}(t)\, dt$$

- Decoder:

$$\hat{X}_i(t) = \sum_{k=1}^{K} z_{ik}\, w_k^{(O)}(t)$$

- Reconstruction loss (continuous):

$$L = \sum_{i=1}^{N_{\text{train}}} \int_T \left(X_i(t) - \hat{X}_i(t)\right)^2 dt.$$

- Impose an **orthonormality constraint** on the functional weights (similar to FPCs):

$$\langle w_k^{(\cdot)}, w_\ell^{(\cdot)} \rangle = \delta_{k\ell}.$$

Connection to FPCA

- In practice, curves are observed at discrete points $\{t_1, \ldots, t_J\}$ with numerical weights ω_j.
- The integral is replaced by a Riemann sum:

$$\int_T X_i(t)\, w_k^{(l)}(t)\, dt \approx \sum_{j=1}^{J} \omega_j\, X_i(t_j)\, w_k^{(l)}(t_j).$$

- The discrete reconstruction loss becomes:

$$L \approx \sum_{i=1}^{N_{\text{train}}} \frac{1}{J} \sum_{j=1}^{J} \left(X_i(t_j) - \hat{X}_i(t_j)\right)^2.$$

- As J grows large, this discrete objective converges to the continuous one.
- With orthonormal constraints on the (discrete) functional weights, the learned weights span (approximately) the same space as FPCAs eigenfunctions.

Connection to AE

- **Faster convergence and better generalization**
  - Feature layer summarizes temporal structure before entering the trainable network.
- **Smooth functional outputs**
  - Decoder reconstructs curves as linear combinations of smooth basis functions.
  - FAE outputs a continuous function, not only point-wise predictions at observed timestamps.
- **Irregular grids are handled naturally**
  - Numerical integration weights are adjusted per curve.
  - AE must deal with missing time points explicitly (e.g., zeros and masking).

Experiment1

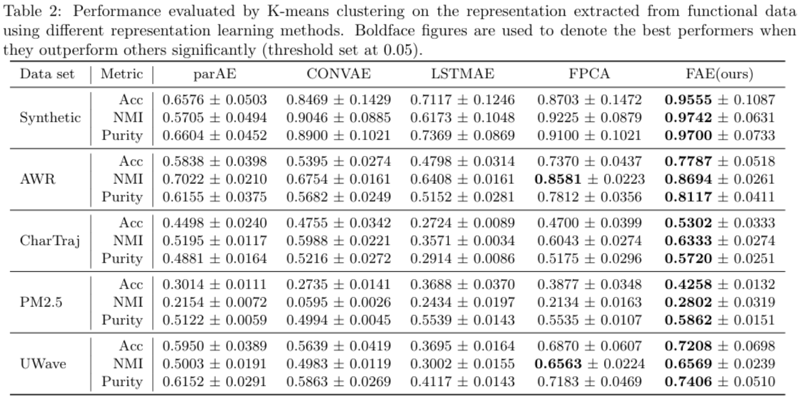
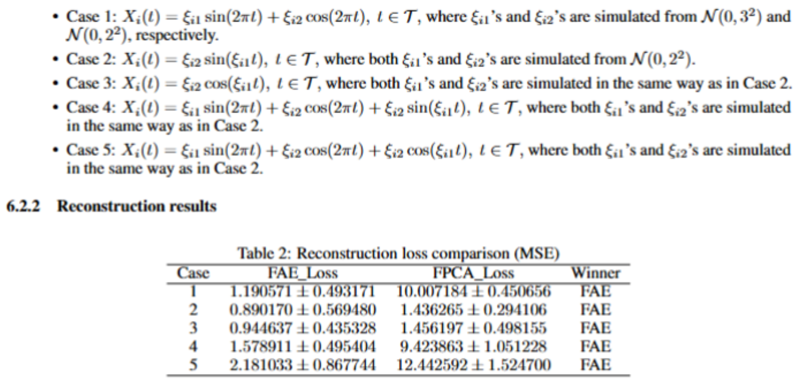


Table 2: Performance evaluated by K-means clustering on the representation extracted from functional data using different representation learning methods. Boldface figures are used to denote the best performers when they outperform others significantly (threshold set at 0.05).

Data set	Metric	parAE	CONVAE	LSTMAE	FPCA	FAE(ours)
Synthetic	Acc	0.6576 ± 0.0503	0.8469 ± 0.1429	0.7117 ± 0.1246	0.8703 ± 0.1472	**0.9555** ± 0.1087
	NMI	0.5705 ± 0.0494	0.9046 ± 0.0885	0.6173 ± 0.1048	0.9225 ± 0.0879	**0.9742** ± 0.0631
	Purity	0.6604 ± 0.0452	0.8900 ± 0.1021	0.7369 ± 0.0869	0.9100 ± 0.1021	**0.9700** ± 0.0733
AWR	Acc	0.5838 ± 0.0398	0.5395 ± 0.0274	0.4798 ± 0.0314	0.7370 ± 0.0437	**0.7787** ± 0.0518
	NMI	0.7022 ± 0.0210	0.6754 ± 0.0161	0.6408 ± 0.0161	**0.8581** ± 0.0223	**0.8694** ± 0.0261
	Purity	0.6155 ± 0.0375	0.5682 ± 0.0249	0.5152 ± 0.0281	0.7812 ± 0.0356	**0.8117** ± 0.0411
CharTraj	Acc	0.4498 ± 0.0240	0.4755 ± 0.0342	0.2724 ± 0.0089	0.4700 ± 0.0399	**0.5302** ± 0.0333
	NMI	0.5195 ± 0.0117	0.5988 ± 0.0221	0.3571 ± 0.0034	0.6043 ± 0.0274	**0.6333** ± 0.0274
	Purity	0.4881 ± 0.0164	0.5216 ± 0.0272	0.2914 ± 0.0086	0.5175 ± 0.0296	**0.5720** ± 0.0251
PM2.5	Acc	0.3014 ± 0.0111	0.2735 ± 0.0141	0.3688 ± 0.0370	0.3877 ± 0.0348	**0.4258** ± 0.0132
	NMI	0.2154 ± 0.0072	0.0595 ± 0.0026	0.2434 ± 0.0197	0.2134 ± 0.0163	**0.2802** ± 0.0319
	Purity	0.5122 ± 0.0059	0.4994 ± 0.0045	0.5539 ± 0.0143	0.5535 ± 0.0107	**0.5862** ± 0.0151
UWave	Acc	0.5950 ± 0.0389	0.5639 ± 0.0419	0.3695 ± 0.0164	0.6870 ± 0.0607	**0.7208** ± 0.0698
	NMI	0.5003 ± 0.0191	0.4983 ± 0.0119	0.3002 ± 0.0155	**0.6563** ± 0.0224	**0.6569** ± 0.0239
	Purity	0.6152 ± 0.0291	0.5863 ± 0.0269	0.4117 ± 0.0143	0.7183 ± 0.0469	**0.7406** ± 0.0510

Figure 3: Experiment

Experiment2:recon



- Case 1: $X_i(t) = \xi_{i1}\sin(2\pi t) + \xi_{i2}\cos(2\pi t),\ t \in \mathcal{T}$, where ξ_{i1}'s and ξ_{i2}'s are simulated from $\mathcal{N}(0, 3^2)$ and $\mathcal{N}(0, 2^2)$, respectively.
- Case 2: $X_i(t) = \xi_{i2}\sin(\xi_{i1}t),\ t \in \mathcal{T}$, where both ξ_{i1}'s and ξ_{i2}'s are simulated from $\mathcal{N}(0, 2^2)$.
- Case 3: $X_i(t) = \xi_{i2}\cos(\xi_{i1}t),\ t \in \mathcal{T}$, where both ξ_{i1}'s and ξ_{i2}'s are simulated in the same way as in Case 2.
- Case 4: $X_i(t) = \xi_{i1}\sin(2\pi t) + \xi_{i2}\cos(2\pi t) + \xi_{i2}\sin(\xi_{i1}t),\ t \in \mathcal{T}$, where both ξ_{i1}'s and ξ_{i2}'s are simulated in the same way as in Case 2.
- Case 5: $X_i(t) = \xi_{i1}\sin(2\pi t) + \xi_{i2}\cos(2\pi t) + \xi_{i2}\cos(\xi_{i1}t),\ t \in \mathcal{T}$, where both ξ_{i1}'s and ξ_{i2}'s are simulated in the same way as in Case 2.

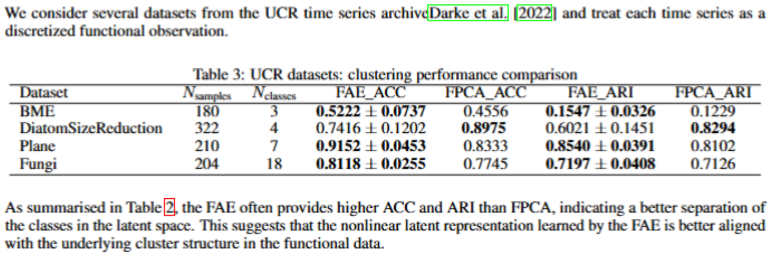
6.2.2 Reconstruction results

Table 2: Reconstruction loss comparison (MSE)

Case	FAE_Loss	FPCA_Loss	Winner
1	1.190571 ± 0.493171	10.007184 ± 0.450656	FAE
2	0.890170 ± 0.569480	1.436265 ± 0.294106	FAE
3	0.944637 ± 0.435328	1.456197 ± 0.498155	FAE
4	1.578911 ± 0.495404	9.423863 ± 1.051228	FAE
5	2.181033 ± 0.867744	12.442592 ± 1.524700	FAE

Figure 4: recon

Experiment2:cluster



We consider several datasets from the UCR time series archiveDarke et al. [2022] and treat each time series as a discretized functional observation.

Table 3: UCR datasets: clustering performance comparison

Dataset	N_{samples}	N_{classes}	FAE_ACC	FPCA_ACC	FAE_ARI	FPCA_ARI
BME	180	3	**0.5222 ± 0.0737**	0.4556	**0.1547 ± 0.0326**	0.1229
DiatomSizeReduction	322	4	0.7416 ± 0.1202	**0.8975**	0.6021 ± 0.1451	**0.8294**
Plane	210	7	**0.9152 ± 0.0453**	0.8333	**0.8540 ± 0.0391**	0.8102
Fungi	204	18	**0.8118 ± 0.0255**	0.7745	**0.7197 ± 0.0408**	0.7126

As summarised in Table 2, the FAE often provides higher ACC and ARI than FPCA, indicating a better separation of the classes in the latent space. This suggests that the nonlinear latent representation learned by the FAE is better aligned with the underlying cluster structure in the functional data.

Figure 5: cluster

Future work

An interesting direction for future work is to extend functional autoencoders so that they simultaneously capture both amplitude and phase variability, for example by jointly learning representations of vertical changes in function values and horizontal time warping within a unified latent functional space.

Reference

[1] S. Wu, C. Beaulac, and J. Cao,
"Functional autoencoder for smoothing and representation learning,"
arXiv preprint arXiv:2401.09499, 2024.

[2] T.-Y. Hsieh, Y. Sun, S. Wang, and V. G. Honavar,
"Functional autoencoders for functional data representation learning,"
in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, Society for Industrial and Applied Mathematics, 2021, pp. 666–674.

[3] R. Zhong, C. Zhang, and J. Zhang,
"Nonlinear functional principal component analysis using neural networks,"
arXiv preprint arXiv:2306.14388, 2023.

Thanks!