skfda.preprocessing.registration.landmark_shift_deltas

skfda.preprocessing.registration.landmark_shift_deltas(*fd*, *landmarks*, *location=None*)
[source]

Returns the corresponding shifts to align the landmarks of the curves.

Let t^* the time where the landmarks of the curves will be aligned, and t_i the location of the landmarks for each curve. The function will calculate the corresponding δ_i shuch that $t_i = t^* + \delta_i$.

This procedure will work independent of the dimension of the domain and the image.

Parameters:

- **fd** (`FData`) – Functional data object.
- **landmarks** (*array_like*) – List with the landmarks of the samples.
- **location** (*numeric or callable, optional*) – Defines where the landmarks will be alligned. If a numer or list is passed the landmarks will be alligned to it. In case of a callable is passed the location will be the result of the the call, the function should be accept as an unique parameter a numpy array with the list of landmarks. By default it will be used as location $\frac{1}{2}(max(\text{landmarks}) + min(\text{landmarks}))$ wich minimizes the max shift.

Returns: Array containing the corresponding shifts.

Return type: `numpy.ndarray`

Raises: `ValueError` – If the list of landmarks does not match with the number of samples.

Examples

```
>>> from skfda.datasets import make_multimodal_landmarks
>>> from skfda.datasets import make_multimodal_samples
>>> from skfda.preprocessing.registration import landmark_shift_deltas
```

We will create a data with landmarks as example

```
>>> fd = make_multimodal_samples(n_samples=3, random_state=1)
>>> landmarks = make_multimodal_landmarks(n_samples=3, random_state=1)
>>> landmarks = landmarks.squeeze()
```

The function will return the corresponding shifts

```
>>> shifts = landmark_shift_deltas(fd, landmarks)
>>> shifts.round(3)
array([ 0.25 , -0.25 , -0.231])
```

The registered samples can be obtained with a shift

```
>>> fd.shift(shifts)
FDataGrid(...)
```