Kernel Embeddings of Longitudinal Data

Darren Shen(✉) and Fabio Ramos

School of Information Technologies, University of Sydney, Sydney, Australia
{darren.shen,fabio.ramos}@sydney.edu.au

Abstract. Longitudinal data is the repeated observations of individuals through time. They often exhibit rich statistical qualities, such as skew or multimodality, that are difficult to capture using traditional parametric methods. To tackle this, we build a non-parametric Markov transition model for longitudinal data. Our approach uses kernel mean embeddings to learn a transition model that can express complex statistical features. We also propose an approximate data subsampling technique based on kernel herding and random Fourier features that allows our method to scale to large longitudinal data sets. We demonstrate our approach on two real world data sets.

1 Introduction

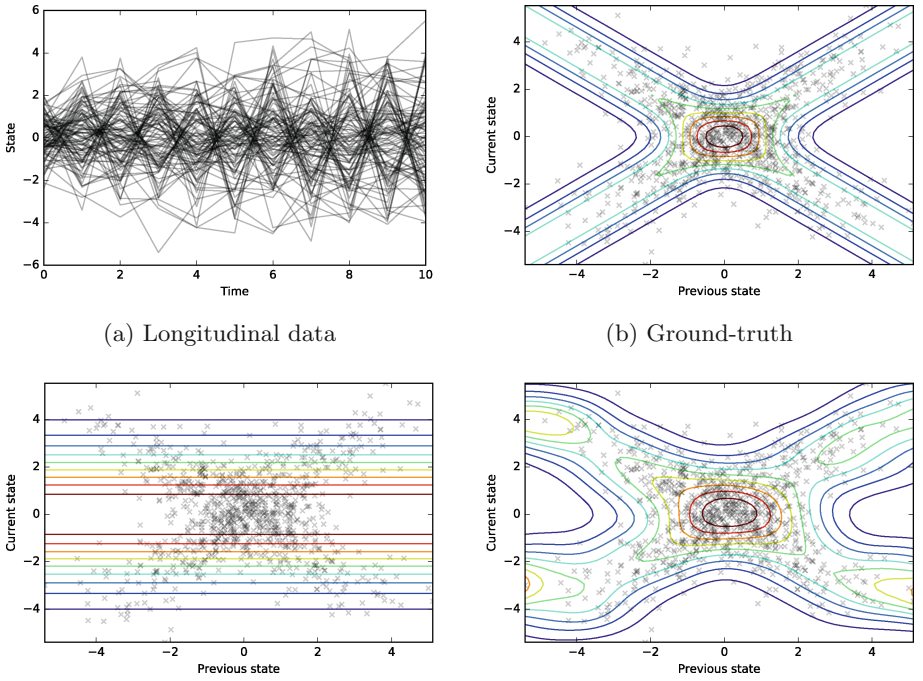
The study of longitudinal data plays an important role in medicine and social sciences. Their defining characteristic is the repeated observation of outcomes for a group of individuals over time. Unlike cross-sectional studies, which only produce a single snapshot in time, longitudinal studies use repeated measurements to produce paths or trajectories through time. These trajectories are invaluable for studying how variables change over time. For example, a longitudinal study of the protein content of cow milk can reveal patterns about the effect of cow diet on milk over time.

Although longitudinal data can be very informative, they require careful statistical treatment because repeated observations from the same individual are often correlated. For example, the dosage of drugs given to patients undergoing medical treatment affects the severity of the disease, which in turn affects the dosage. Ignoring this correlation can lead to invalid inferences, so robust models for longitudinal data need to account for this effect.

One way to capture intra-subject correlation is to use a *transition model*, which describes how an observation relates to past observations. Suppose, for each individual i, we have a sequence of n_i observations $Y_{i1}, \ldots, Y_{in_i}$. A transition model assumes that the j-th observation Y_{ij} of individual i is a function of p previous observations $Y_{ij-1}, \ldots, Y_{ij-p}$ from the same individual. If we model the relationship with past observations as a probability distribution, then we obtain an order-p Markov model for each trajectory:

$$P(Y_{i1}, \ldots, Y_{in_i}) = P(Y_{i1}) \prod_{j=2}^{n_i} P(Y_{ij} \mid Y_{ij-1}, \ldots, Y_{ij-p}). \tag{1}$$

B.H. Kang and Q. Bai (Eds.): AI 2016, LNAI 9992, pp. 495–506, 2016.
DOI: 10.1007/978-3-319-50127-7_42



(a) Longitudinal data

(b) Ground-truth

(c) Gaussian process model

(d) Kernel embeddings model

Fig. 1. Example of longitudinal data where transitions are bimodal. (a) Spaghetti plot of the trajectories. (b) Ground-truth transition model $P(Y_{ij} \mid Y_{ij-1})$ for each value of Y_{ij} (y-axis) and Y_{ij-1} (x-axis). (c) Contour plot of transition model learned using a Gaussian process. (d) Contour plot of transition model learned using our method of kernel embeddings.

Traditional methods for learning the transition model $P(Y_{ij} \mid Y_{ij-1}, \ldots, Y_{ij-p})$ usually make parametric assumptions that restrict the expressiveness of the model. For instance, consider an order-1 Markov transition model with real-valued observations $Y_{ij} = \text{sign}(u_{ij})(0.9Y_{ij-1} + v_{ij})$, where $u_{ij}, v_{ij} \sim \mathcal{N}(0, 1)$. Figure 1a shows 100 trajectories generated using this model. Because $\text{sign}(u_{ij})$ flips the sign of the state with 50% chance, the transition process is bimodal (see Fig. 1b). Figure 1c and d compares the transition model learned using a Gaussian process, with the one learned using our approach based on kernel mean embeddings. When we condition on a particular previous state Y_{ij-1} by slicing the contours vertically, we can see that the Gaussian process produces a normal distribution, which cannot account for the bimodality. In fact, the Gaussian process can only explain the bimodality by treating it as observation noise. On the other hand, each vertical slice in our method produces a distribution that faithfully reflects the characteristics of the data.

Our first contribution is a nonparametric Markov model for longitudinal data. By embedding observations into a reproducing kernel Hilbert space, we can learn the transition model from data without parametric assumptions, giving us freedom to model transitions with complex statistical features. We also propose an efficient data subsampling method based on a random Fourier features approximation of the method in [1]. The algorithm selects a representative subset of training points so that training on the subset gives comparable performance to training with the full data set. Our method finds m subsamples in $O(nmD)$ time and $O(nD)$ storage, where D is a parameter. This makes it possible to accurately model large longitudinal data sets without significant computational cost.

2 Related Work

Transition models applied to longitudinal data are typically parametric linear models (such as autoregressive processes). Although these methods can be easily interpreted, they are a poor fit for complex longitudinal data such as the trajectories of vehicles because of nonlinearity or multimodality in the transition process. Most nonparametric methods for longitudinal data such as splines and kernel density estimation[1] do not have strong theoretical guarantees when approximating probability distributions.

A related model is the state-observation model, where observations are generated by a sequence of hidden states. Our setting can be seen as a special case where the observations are noiseless. Several nonparametric methods based on kernel embeddings have been applied to these models [1,3]. In particular, [1] uses subsampling with kernel herding as a way to speed-up their method. The cost of this preprocessing step is, however, quadratic in the number of data points, which makes it infeasible to run on large data sets. Our method is a scalable alternative that focuses specifically on the noiseless observation case.

3 Kernel Embeddings

We briefly review kernel mean embeddings before applying it to longitudinal data. [5] offers a great exposition on the theory and its applications.

3.1 Kernel Mean

A kernel[2] $k_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined on a measurable space $\mathcal{X}$ induces a unique Hilbert space $\mathcal{H}_{\mathcal{X}}$ of functions called a reproducing kernel Hilbert space (RKHS). It is thus named because its inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\mathcal{X}}}$ satisfies the *reproducing property* $f(x) = \langle f, k_{\mathcal{X}}(\cdot, x) \rangle_{\mathcal{H}_{\mathcal{X}}}$ for all $f \in \mathcal{H}_{\mathcal{X}}$ and $x \in \mathcal{X}$, where $k_{\mathcal{X}}(\cdot, x)$ is the function $k_{\mathcal{X}}$ with one of its parameters fixed at x.

[1] KDE is a closely related method, but we only use positive-definite kernels. Without this requirement, we lose all the theoretical benefits discussed in this paper.

[2] A *positive definite kernel* (or just a *kernel*) $k_{\mathcal{X}}$ defined on a measurable space $\mathcal{X}$ satisfies $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k_{\mathcal{X}}(x_i, x_j) \geq 0$ for any $n \in \mathbb{N}$, $c_1, \ldots, c_n \in \mathbb{R}$, and $x_1, \ldots, x_n \in \mathcal{X}$.

Let X be a random variable on $\mathcal{X}$ with distribution $P(X)$. The framework of kernel embeddings revolves around mapping $P(X)$ to its corresponding *kernel mean* in $\mathcal{H}_{\mathcal{X}}$ using:

$$\mu[P(X)] = \mathbb{E}_X[k_{\mathcal{X}}(\cdot, x)] = \int_{\mathcal{X}} k_{\mathcal{X}}(\cdot, x)\, dP(X). \tag{2}$$

A key advantage of this particular mapping is that it is injective for *characteristic* kernels [2]. In other words, all statistical features of the distribution are preserved by this mapping. The most commonly used characteristic kernel is the Gaussian kernel $k_\gamma(x, x') = \exp(-\gamma\|x - x'\|^2)$, where γ is a parameter. We will focus on the Gaussian kernel throughout this paper.

This formulation, however, requires $P(X)$ to be known explicitly, which is rare in practical problems. Suppose we only have samples from $P(X)$ and we wish to estimate $P(X)$. We can do this by estimating its kernel mean $\mu[P(X)]$ instead. Let $\{x_i\}_{i=1}^n$ be n samples drawn i.i.d. from $P(X)$, then the *empirical kernel mean* is defined to be the sample average,

$$\widehat{\mu}[P(X)] = \frac{1}{n}\sum_{i=1}^{n} k_{\mathcal{X}}(\cdot, x_i). \tag{3}$$

If the Rademacher complexity of $P(X)$ and $\mathcal{H}_{\mathcal{X}}$ is bounded by $O(n^{-1/2})$, the distance $\|\widehat{\mu}[P(X)] - \mu[P(X)]\|$ in the RKHS $\mathcal{H}_{\mathcal{X}}$ converges to zero at a rate of $O(n^{-1/2})$ with high probability [2]. Since this convergence rate is independent of the dimensionality of $\mathcal{X}$, kernel embeddings works well in high dimensions.

3.2 Conditional Kernel Mean

We now focus on the kernel mean of conditional distributions, which forms the foundation of our method. Let X and Y be random variables over measurable spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively. Given kernels $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ on $\mathcal{X}$ and $\mathcal{Y}$, respectively, the embedding of $\mu[P(Y \mid x)]$ is defined as,

$$\mu[P(Y \mid x)] = \mathbb{E}_{Y|x}[k_{\mathcal{Y}}(\cdot, y)] = \int_{\mathcal{Y}} k_{\mathcal{Y}}(\cdot, y)\, dP(Y \mid x). \tag{4}$$

Given i.i.d. samples $\{(x_i, y_i)\}_{i=1}^n$ from the joint distribution $P(X, Y)$, the empirical estimate of Eq. 4 is given by the weighted sum [3]:

$$\widehat{\mu}[P(Y \mid x)] = \sum_{i=1}^{n} w_i k_{\mathcal{Y}}(\cdot, y_i), \tag{5}$$

where $w_i = ((K + n\epsilon_n I_n)^{-1}\mathbf{k}_x)_i$. K is the Gram matrix $(k_{\mathcal{X}}(x_i, x_j)) \in \mathbb{R}^{n\times n}$, I_n is the $n \times n$ identity matrix, $\mathbf{k}_x$ is the vector $(k_{\mathcal{X}}(x, x_i)) \in \mathbb{R}^n$, and ϵ_n is a regularisation parameter to reduce overfitting. The empirical kernel mean of marginal distributions from Eq. 3 is simply a special case of Eq. 5 with uniform weights $w_i = 1/n$. Indeed, non-uniform weights capture the effect of conditioning on $X = x$. When ϵ_n decreases at an appropriate rate as $n \to \infty$, this empirical embedding converges to the true conditional distribution, albeit at a slower rate than the marginal case [3].

4 Kernel Embeddings of Longitudinal Data

We now show how we can apply kernel embeddings specifically to longitudinal data. The mapping between kernel embeddings and our setting is summarised Table 1. As we can see, the mapping is straightforward, but there are some simplifying assumptions, which we discuss below:

1. We use an order-1 Markov model with each observations in $\mathbb{R}^d$. Hence, the model predicts the current observation Y given only the previous observation X. Since both X and Y represent observations, we can use the same space and kernel for both variables. We can extend our discussion to higher order Markov models by setting $\mathcal{X}$ to be the product space of multiple past observations and defining an appropriate kernel over that space.
2. We assume that transition probabilities are independent of time and the individual. This allows us to train a single transition model using all the data. We can relax this assumption by training separate transition models for different time measurements or different groups of individuals (e.g. separate models for male and female). This can reduce the variance of each model, but at the cost of less training data.
3. We focus on the Gaussian kernel $k_\gamma(x, x') = \exp(-\gamma\|x - x'\|^2)$ because it is widely used and has useful theoretical properties in the RKHS. Using other kernels is possible, but we lose some theoretical guarantees.

Table 1. Mapping between the general framework of kernel embeddings in Sect. 3 and our problem setting of longitudinal data.

Kernel embeddings	Our setting	Meaning in our setting
$X \in \mathcal{X}$	$X \in \mathbb{R}^d$	Previous observation
$Y \in \mathcal{Y}$	$Y \in \mathbb{R}^d$	Current observation
$\{(x_i, y_i)\}_{i=1}^n$	$\{(x_i, y_i)\}_{i=1}^n$	Transition model examples
$P(Y \mid X)$	$P(Y \mid X)$	Transition model
$k_\mathcal{X}(x, x')$	$k_\gamma(x, x') = \exp(-\gamma\|x - x'\|^2)$	Kernel on previous observation
$k_\mathcal{Y}(x, x')$	$k_\gamma(x, x') = \exp(-\gamma\|x - x'\|^2)$	Kernel on current observation

4.1 Embedding the Transition Model

To estimate the kernel mean of the transition model $P(Y \mid X)$, we first need to extract training samples from the trajectories. For each trajectory i, we can use consecutive observation pairs $\{(Y_{ij-1}, Y_{ij})\}_{j=2}^{n_i}$ as samples from the transition model. We can then combine the sample pairs from every trajectory to form the full set of training pairs which we denote as $\{(x_i, y_i)\}_{i=1}^n$. If we have t trajectories, then we have in total $n = \sum_{i=1}^t (n_i - 1)$ training examples. Substituting these n observation pairs into Eq. 5 gives us an estimate $\widehat{\mu}[P(Y \mid X)]$ of the transition model in the RKHS.

4.2 Estimation of Embedding Statistics

In order to use the embedding $\widehat{\mu}[P(Y \mid X)]$ for probabilistic inference, we need to extract statistical information from it. Unfortunately, recovering the full conditional distribution that maps to the embedding is difficult (since it may not even exist). Typical methods assume that the distribution is a mixture and solve a quadratic program to find the mixture parameters [4]. These methods are time consuming and require the number of mixture components a-priori.

Fortunately, when we use Gaussian kernels, various statistics of an embedding in the form of Eq. 5 can be estimated by exploiting the reproducing property [6]. Notably, a consistent estimator of the density $p(y_0 \mid x)$ at $Y = y_0$ is given by,

$$p(y_0 \mid x) = \sum_{i=1}^{n} w_i J_{y_0,h}(y_i), \tag{6}$$

where $J_{y_0,h}(\cdot)$ is a Gaussian smoothing kernel centred at y_0 with bandwidth h:

$$J_{y_0,h}(y) = \frac{1}{\pi^{d/2} h^d} \exp(-\|y - y_0\|^2 / h^2). \tag{7}$$

If any of the weights in Eq. 6 are negative, however, the density may be negative, so we may not be able to use this directly. We follow [4] by clipping negative weights at zero and renormalising them to get new weights $w_1^*, \ldots, w_n^*$ so that Eq. 6 forms a valid density:

$$w_i^* = \frac{\max(w_i, 0)}{\sum_{j=1}^{n} \max(w_j, 0)}. \tag{8}$$

Using these new weights may be justified as follows. Assume $y_1, \ldots, y_n$ are unique. As $h \to 0$, $J_{y_0,h}(y)$ becomes the Dirac delta $\delta(y_0 - y)$. When we set y_0 to be a training sample y_k, the right hand side of Eq. 6 reduces to just w_k. This implies that w_k is a consistent estimator of a density, which is nonnegative in the limit. Using a similar argument [1], we also can show that $\sum_{i=1}^{n} w_i$ is a consistent estimator of 1. Hence, in the limit of $n \to \infty$ and $h \to 0$, all weights are nonnegative and sum to 1, so the modified weights are the same as the true weights and we obtain the true density.

4.3 Summary of the Algorithm

The full algorithm takes a set of trajectories as input. We first extract ordered pairs of observations $\{(x_i, y_i)\}_{i=1}^{n}$ from the trajectories as samples from the transition process. We can then use Eq. 5 to get the weights of the predictive embedding conditioned on a given previous observation. To speed up prediction over many different previous observations, we can precompute $\rho = (K + n\epsilon_n I_n)^{-1}$ since it doesn't depend on the previous observation. Then, to make a prediction, we simply compute $\mathbf{k}_x$ and multiply with ρ to get the weights of the predictive embedding. We can then recover the density of the predictive distribution using Eq. 6 with the transformed weights from 8. We can tune the hyperparameters γ and ϵ_n using cross-validation. We set the bandwidth h to be $1/\gamma$. The complete pseudocode is given in Algorithm 1.

Algorithm 1. Learning and prediction of the transition model.

function LEARN-TRANSITION-MODEL(γ, $\{y_{1j}\}_{j=1}^{n_1}, \ldots, \{y_{tj}\}_{j=1}^{n_t}$)
  Extract training pairs $\{(x_i, y_i)\}_{i=1}^{n}$ from trajectories.
  Compute Gram matrix $K \in \mathbb{R}^{n \times n}$ where $K_{ij} = \exp(-\gamma \|x_i - x_j\|^2)$.
  return $\{(x_i, y_i)\}_{i=1}^{n}$ and $\rho = (K + n\epsilon_n I_n)^{-1} \in \mathbb{R}^{n \times n}$.
end function

function PREDICT-TRANSITION-MODEL(γ, $\{(x_i, y_i)\}_{i=1}^{n}$, ρ, x)
  Compute $\mathbf{k}_x \in \mathbb{R}^n$ where the i-th entry is $\exp(-\gamma \|x - x_i\|^2)$.
  Compute prediction weights $\mathbf{w} = (w_1, \ldots, w_n) = \rho \mathbf{k}_x \in \mathbb{R}^n$.
  return new weights $w_i^* = \max(w_i, 0) / \sum_{j=1}^{n} \max(w_j, 0)$.
end function

5 Reducing the Computational Cost

The bottleneck of our method is computing the inverse $\rho = (K + n\epsilon I_n)^{-1}$, which takes $O(n^3)$ naively. We reduce this cost in two ways: a Nyström approximation of K, and a data subsampling method based on kernel herding [1].

5.1 Nyström Approximation

A common technique to compute ρ faster is to approximate K with a low rank matrix. The Nyström method first samples r data points $\widehat{x}_1, \ldots, \widehat{x}_r \in \mathcal{X}$ and then approximates K with the matrix $\tilde{K}_r = C W_r^{\dagger} C^{\top}$, where $C = k_\gamma(x_i, \widehat{x}_j) \in \mathbb{R}^{n \times r}$, $W_r = k_{\mathcal{X}}(\widehat{x}_i, \widehat{x}_j) \in \mathbb{R}^{r \times r}$ and $W_r^{\dagger}$ denotes the pseudo-inverse of W_r. The samples $\widehat{x}_1, \ldots, \widehat{x}_r$ can be selected in many ways. We found that running k-means on the data points and using the cluster centers as the samples worked very well. Once we compute $\tilde{K}$, we can use the Woodbury identity to approximate ρ as,

$$\rho = \frac{1}{n\epsilon_n} \left(I_n - C(n\epsilon_n I_r + W_r^{\dagger} C^{\top} C)^{-1} W_r^{\dagger} C^{\top} \right) \in \mathbb{R}^{n \times n}, \tag{9}$$

where the matrix inversion in this expression is on a $r \times r$ matrix, rather than a $n \times n$ matrix. This reduces the cost of computing the weights to $O(n^2 r)$.

5.2 Subsampling with Kernel Herding

Even with the Nyström approximation, our algorithm still takes quadratic time in the number of data points, which makes it difficult to scale to large longitudinal data sets. A typical technique for reducing computational cost is to only train on a small random subset of the data. The problem of this approach, however, is that it is "too random", so the subset is unlikely to be representative of the original data.

A better approach is to pick representative data points that preserves the "information" in the original data. Since we are working in a RKHS, a natural definition of "information" is the joint embedding of the data in the RKHS [1].

Like before, let X and Y be random variables on $\mathbb{R}^d$ with a common Gaussian kernel $k_\gamma(\cdot,\cdot)$. The empirical joint embedding of samples $\{(x_i, y_i)\}_{i=1}^n$ from $P(X,Y)$ is given by,

$$\widehat{\mu}[P(X,Y)] = \frac{1}{n}\sum_{i=1}^{n} k^{(\times)}((\cdot,\cdot),(x_i,y_i)), \tag{10}$$

where $k^{(\times)}((\cdot,\cdot),(x_i,y_i))$ is the product $k_\gamma(\cdot,x_i)k_\gamma(\cdot,y_i)$. In other words, we wish to find a subset $\{(\bar{x}_p,\bar{y}_p)\}_{p=1}^m$ of the training data such that the empirical joint embedding of the subset is close to that of the original. Let $s_i = \sum_{j=1}^n k^{(\times)}((x_i,y_i),(x_j,y_j))$ and $\bar{s}_{ip} = \sum_{j=1}^{p-1} k^{(\times)}((x_i,y_i),(\bar{x}_j,\bar{y}_j))$. A greedy approach to pick each subsample was devised in [1]:

$$(\bar{x}_1,\bar{y}_1) = \underset{(x_i,y_i)\in\mathcal{D}}{\arg\max}\ \frac{1}{n}s_i \tag{11}$$

$$(\bar{x}_p,\bar{y}_p) = \underset{(x_i,y_i)\in\mathcal{D}}{\arg\max}\ \frac{1}{n}s_i - \frac{1}{p}\bar{s}_{ip}. \tag{12}$$

Intuitively, the term s_i favours samples that are representative of the full data set, while $\bar{s}_{ip}$ penalises samples that are too close to previous subsamples. Obtaining m subsamples takes $O(n^2m)$ time. Unfortunately, this is not very useful since subsampling has the same cost as running Nyström on the full data.

The main bottleneck of this algorithm is computing s_i, which is a $O(n)$ summation. Using random Fourier features [7], we can approximate s_i in $O(D)$ time, where D is the number of random features. For a Gaussian kernel $k_\gamma(\cdot,\cdot)$, there exists a random feature map $\mathbf{z}_d : \mathbb{R}^d \to \mathbb{R}^D$ such that the dot product $\langle \mathbf{z}_d(x), \mathbf{z}_d(x')\rangle$ converges to $k_\gamma(x,x')$ when D is large:

$$\mathbf{z}_d(x) = \sqrt{\frac{2}{D}}[\cos(\boldsymbol{\omega}_1^\top x + b_1),\ldots,\cos(\boldsymbol{\omega}_D^\top x + b_D)]^\top, \tag{13}$$

where $\boldsymbol{\omega}_1,\ldots,\boldsymbol{\omega}_D \in \mathbb{R}^D$ are vectors with entries drawn from $\mathcal{N}(0,2\gamma)$ and $b_1,\ldots,b_D \in \mathbb{R}$ are drawn uniformly on $[0,2\pi)$.

To approximate s_i, we need a random feature map for $k^{(\times)}$. Since this kernel is the product of two Gaussian kernels, $k^{(\times)}((x_i,y_i),(x_j,y_j))$ is just the Gaussian kernel $k_\gamma([x_i;y_i],[x_j;y_j])$ on $\mathbb{R}^{2d}$ where $[x_i;y_i] \in \mathbb{R}^{2d}$ is the concatenation of x_i and y_i. We can then use a neat trick to approximate s_i:

$$s_i = \sum_{j=1}^{n} k_\gamma([x_i;y_i],[x_j;y_j]) \tag{14}$$

$$\approx \sum_{j=1}^{n} \langle \mathbf{z}_{2d}([x_i;y_i]), \mathbf{z}_{2d}([x_j;y_j])\rangle \tag{15}$$

$$= \langle \mathbf{z}_{2d}([x_i;y_i]), \sum_{j=1}^{n} \mathbf{z}_{2d}([x_j;y_j])\rangle \tag{16}$$

$$\triangleq \langle \mathbf{z}^{(i)}, \mathbf{u}\rangle. \tag{17}$$

Algorithm 2. Kernel Herding Subsampling using Random Features.

function FAST-HERDING-SUBSAMPLE(γ, $\{(x_i, y_i)\}_{i=1}^n$)
  Compute feature maps $\mathbf{z}^{(i)} \in \mathbb{R}^D$ for all i using Eq. 13.
  Compute the mean feature map: $\mathbf{u} = \sum_{i=1}^n \mathbf{z}^{(i)} \in \mathbb{R}^D$.
  Initialise $\bar{\mathbf{u}} \in \mathbb{R}^D$ be the zero vector.
  for $p = 1$ to m **do**
    $(\bar{x}_p, \bar{y}_p) = \underset{(x_i, y_i) \in \mathcal{D}}{\arg\max} \frac{1}{n} \langle \mathbf{z}^{(i)}, \mathbf{u} \rangle - \frac{1}{p} \langle \mathbf{z}^{(i)}, \bar{\mathbf{u}} \rangle$.
    Update $\bar{\mathbf{u}} \leftarrow \bar{\mathbf{u}} + \mathbf{z}_{2d}([\bar{x}_p; \bar{y}_p])$.
  end for
  return $\{(\bar{x}_p, \bar{y}_p)\}_{p=1}^m$.
end function

The key advantage of this approximation is that we only need to compute $\mathbf{z}^{(i)}$ and $\mathbf{u}$ once, so s_i can be computed with a $O(D)$ dot product instead of a $O(n)$ sum. The same idea applies to $\bar{s}_{ip}$. The pseudocode is given in Algorithm 2.

6 Experiments

We tested our algorithm in two real world data sets against other probabilistic methods that can learn a transition model. Both data sets have non-Gaussian transition models. To measure the performance of each algorithm, we first extracted pairs of observations from every trajectory and then performed 5-fold cross-validation on all the observation pairs. Hence, each algorithm uses a subset of the data to learn a transition model, which is then tested on unseen data.

For each test fold, we measured the mean negative log-likelihood (NLL) of the unseen data. For a set of test pairs $\{(x_i, y_i)\}_{i=1}^n$, the mean NLL is given by $-\frac{1}{n} \sum_{i=1}^n \log p(y_i \mid x_i)$. The lower this is, the more accurately the method generalises to test data.

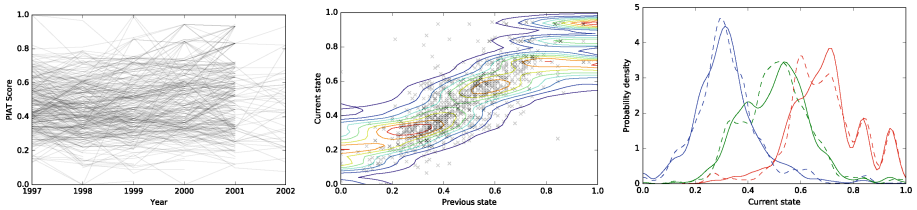
6.1 NLSY97 PIAT Data

We extracted a subset of the National Longitudinal Study on Youths (NLSY97) data set containing annual results on a scholastic achievement test results called the Peabody Individual Achievement Test (PIAT). The data set contains test results for around 6000 individuals from 1997–2002. There were plenty of missing data, so the number of training pairs was only around 6000. This effect was prominent in the year 2002, where the number of data points was less than half of the previous year. We normalised the scores to be between 0 and 1.

We tested our kernel embeddings approach without any approximation (KME-EXACT). We also tested the approximate method (KME-APPROX) by first selecting $m = 200$ subsamples from kernel herding with $D = 50$ random features, and then selecting $r = 100$ K-means cluster centres for the Nyström approximation. We tuned the hyperparameters by maximising the likelihood from cross-validation. More specifically, for each value $\epsilon_n \in \{1.0, 0.1, 0.01\}$, we

Table 2. Cross-validation results for the PIAT data set. The 1st interval counts the % of test points that lie between the 0th and 25th percentile, and so on.

Method	NLL	1st (%)	2nd (%)	3rd (%)	4th (%)
AR(1)	-0.67 ± 0.04	17.6 ± 1.6	34.0 ± 0.9	28.5 ± 0.8	19.9 ± 1.1
GP	-0.70 ± 0.03	18.6 ± 2.1	34.8 ± 1.8	27.7 ± 1.0	19.0 ± 1.3
KME-Exact	-0.87 ± 0.03	22.1 ± 1.9	28.2 ± 1.4	27.4 ± 0.8	22.3 ± 1.2
KME-Approx	-0.74 ± 0.04	19.4 ± 2.3	32.7 ± 1.6	28.4 ± 1.9	19.5 ± 1.5



(a) PIAT trajectories (b) Transition model (c) Predictive distributions

Fig. 2. (a) Trajectories in the PIAT data set. (b) Contour of the transition model learned using our exact method on a 2000 data point subset of the PIAT data. (c) The predictive distribution conditioned on three different previous observations $X = 0.25, 0.5, 0.75$ (i.e. vertical slices of the contour). The solid and dashed line shows the densities of our predictive distribution using the exact and approximate methods, respectively.

used a 1D optimiser to tune γ. We found that this worked better than using 2D optimisation on both parameters with numerical gradients.

To compare, we trained an order-1 autoregressive model using ordinary least squares. We also trained a Gaussian process with a RBF kernel using the `GPy` library [9]. We optimised the hyperparameters (kernel variance and lengthscale, noise variance) by maximising the marginal likelihood of the data.

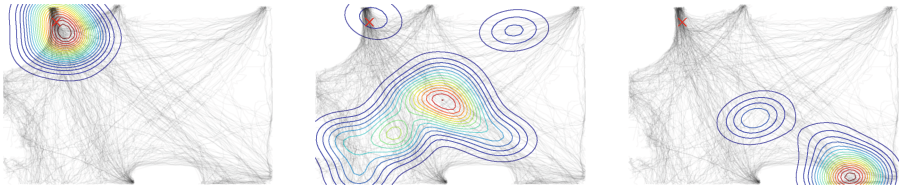
For each algorithm, we obtained their mean NLL on cross-validation as previously described. We also divided the predictive distributions of each method into four intervals that each have 25% of the total probability mass (i.e. the first interval is the mass between the 0th and 25th percentile, and so on). We then counted how many of the unseen points lie in each interval. The closer each of the four counts are as close to 25%, the better the predictive distribution matches the shape of the data. Table 2 reports the results for this data set.

In terms of log-likelihood, kernel mean embeddings outperformed AR(1) and GP. AR(1) performed worse than GP because it is a linear classifier, so it not as expressive as Gaussian processes which are nonlinear. KME-Approx was comparable to GP, even though it used only 200 subsamples out of 6000 data points. Looking at the percentile interval results, we can see that both AR(1) and GP overconcentrated their mass in the middle because it is limited to a normal distribution. Our method had a similar concentration of mass, but to a lesser degree because it is not constrained by any parametric form (see Fig. 2c).

6.2 Edinburgh Pedestrian Data

This data set contains the 2D trajectories of pedestrians walking through an area in the University of Edinburgh [8]. We used trajectories from a single day (Aug 24), which had 664 trajectories and 76,260 data points. The movements of pedestrians are inherently Gaussian, which is not particularly interesting. Hence, for each trajectory i, we used observation pairs $\{(Y_{ij-k}, Y_{ij})\}_{j=k+1}^{n_i}$ that are k frames apart as data samples. This changed the task to modelling the position of a pedestrian after k frames, which exhibits more interesting statistical features like multimodality (see Fig. 3 for the effect of different k on our predictive distribution). We set $k = 10$, which gave us around 70,000 training points. Like the PIAT data set, we normalised the X and Y axes to be between 0 to 1.

Because the data set was so large, we could not use KME-Exact due to memory constraints. We set $m = 500$, $D = 50$ and $r = 100$ for KME-Approx. We also could not use exact GP inference, so we instead used sparse variational GP (SGP) from GPy with a RBF kernel and two outputs to obtain the 2D position. We used 100 inducing points to match the rank of the Nyström approximation in KME-Approx, initialised using K-means cluster centres. We optimised both the hyperparameters and the inducing point locations. Table 3 shows the mean NLL results for 5-fold cross validation on this data set.



(a) k=10 (b) k=50 (c) k=100

Fig. 3. Predictive distributions for the Pedestrian data set with different values of k. (a) Prediction for the location of a pedestrian $k = 10$ frames after, if the pedestrian started at the top left corner. (b) and (c) Predictions with larger k to predict farther into the future.

Table 3. Cross-validation results for the Pedestrian data set.

Method	NLL
KME-Approx	-6.00 ± 0.05
SGP	-1.44 ± 0.01

7 Conclusion

We described a nonparametric Markov model for longitudinal data where the transition model is learned using kernel mean embeddings. This allows us to

model complex longitudinal data where the transition process exhibits statistical features such as skew and multimodality. We also proposed a random features approximation of the data subsampling method from [1], reducing the running time from $O(n^2m)$ to $O(nmD)$. This subsampling method can be used to speed up a wide variety of inference algorithms based on kernel mean embeddings.

For future work, we would like to incorporate explanatory variables by considering the distribution $P(Y_{ij} \mid Y_{ij-1}, x_{ij})$, where x_{ij} is a vector of variables that may influence the observation Y_{ij}. An interesting research direction would be to see if we can combine parametric models for explanatory variables with non-parametric transition models. We would also like to see the effect of using higher order Markov models on the performance of these methods.

References

1. Kanagawa, M., Nishiyama, Y., Gretton, A., Fukumizu, K.: Filtering with state-observation examples via kernel Monte Carlo filter. Neural Comput. **28**(2), 382–444 (2014)
2. Smola, A., Gretton, A., Song, L., Schölkopf, B.: A Hilbert space embedding for distributions. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 13–31. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75225-7_5
3. Song, L., Huang, J., Smola, A., Fukumizu, K.: Hilbert space embeddings of conditional distributions with applications to dynamical systems. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 961–968. ACM, June 2009
4. McCalman, L.R.: Function embeddings for multi-modal Bayesian inference (2013)
5. Muandet, K., Fukumizu, K., Sriperumbudur, B., Schlkopf, B.: Kernel mean embedding of distributions: a review and beyonds. arXiv preprint arXiv:1605.09522 (2016)
6. Kanagawa, M., Fukumizu, K.: Recovering distributions from Gaussian RKHS embeddings. In: AISTATS, pp. 457–465 (2014)
7. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Advances in Neural Information Processing Systems, pp. 1177–1184 (2007)
8. Majecka, B.: Statistical models of pedestrian behaviour in the forum. Master's thesis, School of Informatics, University of Edinburgh (2009)
9. GPy: GPy: a Gaussian process framework in python. http://github.com/SheffieldML/GPy