Differentiable Nonparametric Regression for Multivariate Time Series via Convolution

Narges

David

Abstract

-Rewrite- Nonparametric methods have relied on parametric kernels and cross validation to recover the kernel family and optimal parameters. This severely limits the power of these methods in kernel density estimation, regression, and any consequent tasks. In this paper, we provide a simple formulation of nonparametric regression based on convolution, which enables us to learn the kernel function and parameters directly from temporal data. We show how learning kernels improves the quality of density estimation and regression, compared to traditional cross validation within parametric families of kernels. Moreover, we show our method can easily extend to multivariate input with structural and temporal dependencies. We compare to deep Gaussian Processes, and temporal autoencoders, and bidirectional LSTM for imputing missing values in multivariate time series.

1   Introduction

rewrite- In several areas of data science and machine learning, tractability of computations has often been traded off with simplifying assumption on the dependencies and distribution of the random variables of the model.[] [] [] [] As the size of datasets are growing, these assumptions begin to hurt more than they help [] [] []. Kernel density estimation[] is one of the methods which does not impose most of the simplifying assumptions on the distribution of the data[], and has been previously studied in the context of density estimation, regression, causal inference, conditional distribution modeling, among others. Nonparametric regression[] in particular, is a general method for estimating any function, given samples from the true distribution and a kernel similarity function.

While the theory of kernel regression allows the use of any positive semi definite kernel, in practice, methods only cross-validate over a few well-known kernel functions such as Radial basis(Gaussian), Laplace, or other simple kernels, and fail to consider the entire space of legal kernels. Of course when one leaves this step to cross-validation nothing more is reasonable to expect. In this paper, we try to liberate the kernel regression method from searching within a set of pre-defined kernel families.

In particular, we focus on irregularly measured temporal time series, and show how to construct a differentiable formulation of kernel density estimation for single time series as well as multiple dependent time series. Unlike most other multivariate models, our method can handle asynchronous observations of different variables across time. Our method is limited to discrete time observation, and we use a differentiable formalization and gradient descent to estimate the kernel function over its domain. As we show, not only this method is easily able to recover multivariate kernels, it also runs efficiently on large datasets.

This paper is motivated by medical domain, where measurements of different biomarkers in our bodies are often performed asynchronously and measurements are often sparse in time. Models which impute the unobserved values between measurements can help discover abnormal temporal

patterns and diagnose unknown diseases before late stage symptoms develop. We evaluate our method on a public temporal dataset (physionet challenge) and use the

2 Univatiate kernel regression: Learning the Kernel

Imagine the input to be samples from D time series, each sampled irregularly. An example context would be different types of lab measurements for a patient at different time points across their life. We denote the samples as $x_{t_1^1}^1, x_{t_2^1}^1, ..., x_{t_{n_1}^1}^1, ..., x_{t_1^D}^D, x_{t_2^D}^D, ..., x_{t_{n_D}^D}^D$, where x^d refers to time series d and $t_1^d, ... t_{n_d}^d$ refer to the time points over which time series d is sampled.

Kernel regression provides a general formalism for estimating any function with additive noise. Let's start from a single time series, $x(t)$. Kernel regression assumes the following.

$$x = f(t) + \epsilon$$

$$\epsilon \sim N(0, \sigma^2)$$

Given observed samples $x_{t_1}, ... x_{t_n}$ from the series, general function regression with additive noise lets us estimate the value of x at a new time point t_{new} as follows.

$$x(t_{new}) = \mathbf{E}_{x \sim P(x|t=t_{new})}[x]$$

$$\mathbf{E}_{x \sim P(x|t=t_{new})}[x] = \int_x x P(x|t = t_{new}) dx = \int_x x \frac{P(x, t = t_{new})}{P(t_{new})} dx$$

At this point, one can use kernel density estimation to estimate the probabilities $P(x, t = t_{new})$ and $P(t_{new})$ from the training data. Nadaraya[] and Watson[]showed that using a positive semidefinite kernel function $K(t, t')$, the nonparametric regression formulation is reduced to:

$$\mathbf{E}_{x \sim P(x|t=t_{new})}[x] = \frac{\sum_{i=1}^n x_{t_i} K(t_{new}, t_i)}{\sum_{i=1}^n K(t_{new}, t_i)} \tag{1}$$

We can now rewrite the nonparametric regression using convolution operator. To be able to use functional notation, we first write the sequence of observed samples $x_{t_1}, ..., x_{t_n}$ as a function: $\bar{X}_{train}(t)$ $= \sum_{i=1}^n x_{t_i} \delta(t, t_i)$, where $\delta(t, \tau_0) = 1$ when $t = \tau_0$, and 0 otherwise.

Denoting convolution operator as $*$, i.e. $(k * f)(t) = \int_\tau K(t - \tau) f(\tau) d\tau$, the numerator of the kernel regression is equal to: $\sum_{i=1}^n x_{t_i} K(t_i - t_{new}) = (K * \bar{X}_{train})(t_{new})$. The denominator $P(t_{new})$ can similarly be written as a convolution of the kernel function with a sequence of 1s at each point at which we have a sample, denoted as $I(\bar{X}_{train} : observed)(t) = \sum_{i=1}^n \delta(t, t_i)$.

$$\sum_{i=1}^n K(t_{new}, t_i) = (K * I(\bar{X}_{train} : observed))(t_{new})$$

So the kernel regression formulation of Nadaraya and Watson reduces to the following formulation.

$$\mathbf{E}_{x \sim P(x|t=t_{new})}[x] = \frac{(K * \bar{X}_{train})(t_{new})}{(K * I(\bar{X}_{train} : observed))(t_{new})}$$

This formulation has already been used in image processing literature[?,]nd [?, ?,] although with parametric kernels. The benefit of this formulation is that we have a differentiable formulation of the kernel regression. Therefore we can learn $K(\tau)$ at each position τ within the kernel domain, using function composition and back-propagation[]. We can also compose this differentiable kernel regression module within any subsequent differentiable operators and perform multiple tasks. In this paper we use leave-one-out imputation mean squared error as the loss function. In practice, we assume the domain of K is bounded between $[-M, M]$, therefore the learning task will have $2M + 1$ parameters.
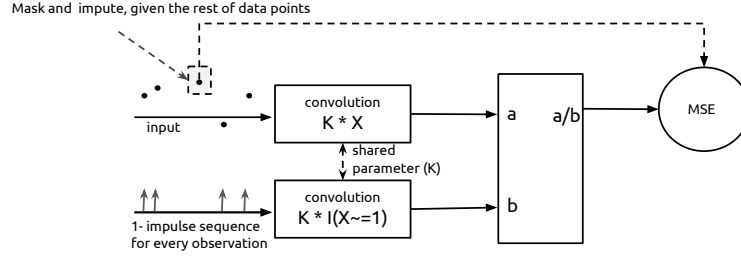
Figure 1 shows the architecture of this model.

Figure 1: Architecture for end-to-end differentiable univariate kernel regression.

3 Multivariate kernel regression: Learning the Temporal Kernel and Dependency structure

Let's assume that we have D time series now, where time series are sampled assynchronously. Traditional multivatiate kernel methods can not handle this case, because the kernels are defined over observations of all variables at each time point. Additionally, most traditional kernel methods assume the kernel to be the product of independent kernels on each dimension therefore missing the correlation structure between variables.

We provide two extensions of our univariate formulation of kernel regression here.

3.1 Shallow Model: Linear Multivatiate Kernel

In our first extension, we can extend the kernel K to a matrix of size $D \times (2M+1)$, and optimize each kernel value between series i at time r and series j at time s. The formulation of the kernel regression then becomes a 2D convolution of this kernel matrix with all time series' observed points in the numerator, normalized by the 2D convolution of the kernel matrix with a binary matrix encoding which series at which time point does have a nonzero observation.

$$\mathbf{E}_{x^d \sim P(x^d|t=t_{new})}[x^d] = \frac{(K * \bar{X}_{train}^{1..d})(t_{new})}{(K * I(\bar{X}_{train}^{1..d} : observed))(t_{new})}$$

This formulation allows every variable at time t to be a linear function of all other variables between $t - M$ and $t + M$. An example shown in the experiments section includes a lab value that is a ratio of two other lab values by definition. If we hadn't known this in advance and applied multivariate kernel regression, based on this formualtion, the lab value is approximated with the difference of the two observations instead of the ratio. In practice, however, for many settings this model is sufficient and has strong empirical performance as we will show in experiments section.

Figure 2 on the left shows the model for one output variable. When learning models for multiple time series, each variable would learn a separate kernel over the full set of time series.

3.2 Deep Model: Nonlinear Multivatiate Kernel

In out next extension, we define the imputation function to be a general function of univariate kernel regressions.

$$x^d(t_{new}) = f(\frac{k^1 * x^1}{k^1 * I^1}(t_{new}), \frac{k^2 * x^2}{k^2 * I^2}(t_{new}) \ldots, \frac{k^D * x^D}{k^D * I^D})(t_{new}))$$

This formulation allows us to share the univariate kernels $k^1, .. k^D$ between different outputs, and lowers the number of paramameters from $D \times D \times (2M + 1)$ to $D \times (2M + 1) + D \times P$ where P is the number of parameters encoding the nonlinear multivatiate interactions function f.

Figure 2 on the right is an architecture for this model. This model allows us to handle the case where variables are measured anytime between $t - M$ to $t + M$. However, when a variable is never observed in that period, univariate kernel regression is not defined. As we describe, the construction of f should allow inference despite at the occurance of missing values.
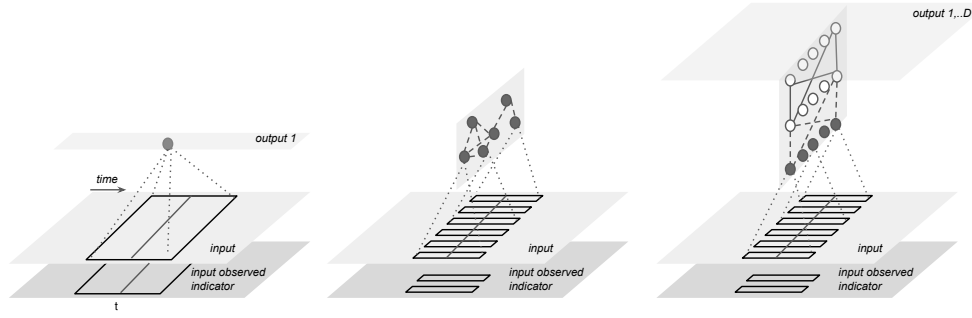
Figure 2: Multivariate kernel regression networks. The architecture on the left is the shallow linear kernel regression. The grey node in that network is a mini-network which takes the input and observation mask, convolves it with the kernel, and returns the division of the two values. The architecture in the middle shows how multiple variables with arbitrary dependencies can be approximated in two stages. The architecture on the right is the unraveled version of middle architecture, in which each time series calculates a univariate estimate from the observations, and a feedforward network with D nodes at the first and last layers applies nonparametric multivatiate inference. Minimizing the mean squared error of each output node D at time t will concurrently train multivatiate nonparametric inference network and the univariate kernels for each time series.

We desire the combination funciton f to return the MAP estimate of variable d at time t_{new}, given the neighboring observations. If we model each univariate kernel regression as a MAP estimate on the univariate observations using nonparametric density estimation, what we want is the MAP inference over a probabilistic graphical model over the input variables. In this graphical model, node potentials are defined by the univariate kernel density estimator, and depenency structure and forms of the 2-nd(or higher) degree potentials are unknown. Many successful nonparametric multivariate density estimators for continuous variables already exist, including RNADE[], DARN[], MADE[], Deep Directed Generative Autoencodes[], Reconstruction Contractive Autoencoder(RCAE)[]http://arxiv.org/pdf/1211.4246v5.pdf and related models [][][].

We are only interested in the models with very fast training and inference and high accuracy. [] has shown that denoising autoencoder with Gaussian noise and squared error loss function, with the penalty term on the norm of the reconstruction with respect to the input can be a nonparametric density estimator for continuous multivariate models.

In our model, we use this nonparametric density estimator on top of the univariate kernel estimations to learn a multivariate nonparametric model for our asynchronous timeseries.

4   Related Work

traditional   kernel   methods,   GP,   univariat2e   and   multivariate.   (For   GP) http://www.cs.toronto.edu/ rgrosse/icml2013-gp.pdf code: https://github.com/jamesrobertlloyd/gp-structure-search

5   Convoutional Neural Netowrks on Graphs with Unknown Structure

6   Experiments and Results

6.1   Data

We normalize each series to be zero mean, unit variance (each measurement per person normalized on its own).. more details later.. Training data is 5K, and validation set is another 5K. There are 100 lab measurements over 109 months.

6.2 Synthetic Data

Sampling multiple codependent Gaussian processe (I know how to do it) How about other forms of dependencies??

6.3 Data Augmentation for Training

In order to successfully train the model with finite training data, we augmented each time series by adding Gaussian noise to each observation, and wiggling the time of each observation by a small amount, randomly drawn from another Gaussian distribution with standard deviation of 1. This step is very important for convergence of the method.

6.4 Technical Details of the implementation

In order for the back-propagation to remain stable, we limited the derivative of the ratio such that if the magnitude of denominator was smaller than an epsilon, we did not back-propagate its derivative. We also filtered the gradients within range of [-10 +10], as higher values would potentially cause numerical instability. The effect of the limit was that the convergence would be slower, but more stable.

7 References