A Gaussian Process Approach to Forecast Evapotranspiration

Ehsan Kourkchi[a,b,*], et al.

[a]*Department of Civil and Environmental Engineering and Water Resources Research Center, University of Hawaii at Manoa, Honolulu, HI 96822, USA*
[b]*Institute for Astronomy, University of Hawaii, 2680 Woodlawn Drive, Honolulu, HI 96822, USA*

Abstract

In this study, we present a Bayesian machine learning model based on the Gaussian Process Regression (GPR) methodology to forecast the meteorological parameters such as terrestrial evapotranspiration (ET), (and other). We take an auto-regressive approach where each value in the time series is predicted based on a few previous data points. Our model benefits from the Particle Swarm Optimization (PSO) technique to optimize the hyperparameters of the adopted GPR kernel.

Keywords: Evapotranspiration, Gaussian Process Regression, Time Series, Forecasting, Particle Swarm Optimization

1. Introduction

Evapotranspiration (ET) measures the evaporated water vapor from the land surface to the atmosphere.

< Discussions about how ET is measured, why it is important, and what models have been previously produced to forecast it. >

Karbasi (2017) has suited the capability of GPR and an hybrid wavelet-GPR in the multi step forecasting of daily evapotranspiration, and reported a better performance of the model in the Summer season.

2. The Autoregressive Framework

We treat the daily values of ET as a sequence of data points ordered in time, i.e. $\boldsymbol{y} = \{ET_1, ET_2, \ldots, ET_t\}$. We build our model based on an autoregressive approach, where we attempt to predict the ET value based on the available data in the previous p days. In this framework, the predicted value of ET_t is described as

$$\tilde{y}_t = \widetilde{ET}_t = f_i(\boldsymbol{x}_t^{(i)}) \tag{1}$$

where $\boldsymbol{x}_t^{(i)} = \{ET_{t-i}, ET_{t-i-1}, \ldots, ET_{t-i-p+1}\}$. Here, f_i is the autoregressive function and i is the temporal offset between the target value, ET_t, and the feature vector, $\boldsymbol{x}_t^{(i)}$. According to this formulation, a 1-day ahead forecast is expressed as $\widetilde{ET}_t = f_1(ET_{t-1}, ET_{t-2}, \ldots, ET_{t-p})$. The deviation of the predicted value from the real measurement is $\epsilon_t = y_t - \tilde{y}_t$. Here, ϵ_t is an i.i.d (independently identically distributed) residual that is assumed to follow a Gaussian distribution, i.e. $\epsilon_t \sim N(0, \sigma_e^2)$.

3. Gaussian Process Regression (GPR)

The Gaussian Process Regression (hereafter GPR) is a non-parametric Bayesian approach with a recipe to make predictions by leveraging the available information provided by the neighbouring points in the parameter space (e.g. Gibbs and MacKay, 1997; Rasmussen and Williams, 2006). This method has been successfully applied in various fields of time series analysis such as the forecasting

*Corresponding author
Email address: ekourkchi@gmail.com (Ehsan Kourkchi)

of the energy demand (Wang et al., 2017/07; Blum and Riedmiller, 2013), the short-term wind power (Chen et al., 2014), and modelling of short-term traffic flows (Wang et al., 2018; Guo et al., 2017).

GPR is free of any specific parametrization. It only assumes that the functional form of $f(.)$ is sufficiently smooth and is determined by tuning the hyperparameters of a kernel that is chosen to address the expected behavior of data. In the GPR formalism, f is randomly drawn from a prior multivariate Gaussian distribution

$$\begin{bmatrix} f(\boldsymbol{x_1}) \\ \vdots \\ f(\boldsymbol{x_n}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x_1}) \\ \vdots \\ \mu(\boldsymbol{x_n}) \end{bmatrix}, \begin{bmatrix} \mathcal{K}_{1,1} & \cdots & \mathcal{K}_{1,n} \\ \vdots & \ddots & \vdots \\ \mathcal{K}_{n,1} & \cdots & \mathcal{K}_{n,n} \end{bmatrix} \right), \quad (2)$$

where n is the number of independent drawn data points, with μ and $\mathcal{K}$ representing the mean and covariance functions, respectively. Most often, this formulation is simply written as $f(.) = \mathcal{GP}(\mu(.), \mathcal{K}(.,.))$. Typically, it is assumed that the prior distribution of $f(.)$ has a zero mean.

Hereafter in this section, let $\boldsymbol{y}$ be a $n \times 1$ column vector that holds the target parameters of n data points whose independent features are stored in $\boldsymbol{X}$, i.e. an $n \times p$ feature matrix. For any pair of points represented by $(\boldsymbol{x_a}, y_a)$ and $(\boldsymbol{x_b}, y_b)$, the covariance matrix is described as

$$\mathcal{K}_{ab} = \sigma_a^2 \delta_{ab} + \kappa_{ab} \,, \quad (3)$$

where σ_a is the uncertainty in the y_a measurement, δ is the Kronecker delta ($\delta_{ab} = 1$ if $a = b$ and $\delta_{ab} = 0$ if $a \neq b$) and κ is the GP kernel. The uncertainties of the measurements are not available in this study, therefore we assume $\sigma_a = 0$. Here, we adopt a simple exponential squared kernel function which is defined as

$$\mathcal{K}_{ab} = \sigma_e^2 \delta_{ab} + \sigma_f^2 \, \exp \sum_{i=1}^{p} -\left(\frac{\boldsymbol{x_{ai}} - \boldsymbol{x_{bi}}}{\ell_i} \right)^2 , \quad (4)$$

where σ_e, σ_f, and ℓ_i are the hyperparameters of the chosen kernel, and are determined in the training process.

3.1. GPR: Training

Taking a Bayesian approach, the objective is to find a set of hyperparameters that maximize the conditional probability $\mathcal{P}(\Theta|\mathcal{D})$, with Θ representing the vector of the kernel hyperparameters (i.e. σ_e, σ_f, and $\ell_1, \ldots, \ell_p$). Here, $\mathcal{D}$ stands for the data, $(\boldsymbol{X}, \boldsymbol{y})$. The law of conditional probability states that $\mathcal{P}(\Theta|\mathcal{D}) \propto \mathcal{P}(\mathcal{D}|\Theta)\mathcal{P}(\Theta)$. With no prior knowledge on the distribution of the model hyperparameters, we simply set $\mathcal{P}(\Theta) = 1$. Therefore, the optimization can be achieved by adopting any technique that maximizes the likelihood function which is defined as $\mathcal{L} = \mathcal{P}(\mathcal{D}|\Theta)$. In the GPR framework, "training" is the process to tune the kernel hyperparameters to maximize the likelihood of $\boldsymbol{y}$ to be drawn from the prior distribution, $\mathcal{N}(0, \mathcal{K})$.

For a normal distribution with covariance $\mathcal{K}$, the logarithm of likelihood has the following form

$$\log \mathcal{L}(\boldsymbol{y}) = -\frac{1}{2} \boldsymbol{y}^T \mathcal{K}^{-1} \boldsymbol{y} - \frac{1}{2} \log |\mathcal{K}| - \frac{n}{2} \log(2\pi). \quad (5)$$

The large number of hyperparameters and the size of utilized data points for training increases the complexity of the optimization algorithms, calling for an extensive computational power. We use the *Particle Swarm Optimization* (hereafter PSO) algorithm to explore the hyperparameter space to maximize the likelihood function given in Equation 5). The PSO methodology is introduced in detail in Section 4.

3.2. GPR: Predictions

Following the optimization of the GPR kernel hyperparameters, we need a formalism to predict $\boldsymbol{y}_*$ for a set of m data points. For simplicity we define $\boldsymbol{X}$ and $\boldsymbol{X}_*$ to be the feature matrices of size $n \times p$ and $m \times p$, respectively. Gaussian process assumes that the joint probability of $\boldsymbol{y}$ and $\boldsymbol{y}_*$ follow the same Gaussian distribution, therefore $(\boldsymbol{y}, \boldsymbol{y}_*|\boldsymbol{X}, \boldsymbol{X}_*) \sim \mathcal{N}(0, \mathcal{K}_+)$. Here, $\mathcal{K}_+$ is expressed as

$$\mathcal{K}_+ = \begin{pmatrix} \mathcal{K}(\mathbf{X}, \mathbf{X}) & \mathcal{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathcal{K}(\mathbf{X}_*, \mathbf{X}) & \mathcal{K}(\mathbf{X}_*, \mathbf{X}_*) \end{pmatrix}, \quad (6)$$

where $\mathcal{K}(\mathbf{X}, \mathbf{X})$ is $n \times n$, $\mathcal{K}(\mathbf{X}_*, \mathbf{X}_*)$ is $m \times m$, $\mathcal{K}(\mathbf{X}, \mathbf{X}_*)$ is $n \times m$ and equals $\left(\mathcal{K}(\mathbf{X}_*, \mathbf{X})\right)^T$.

The goal is to find the distribution of the predicted values, $\boldsymbol{y}_*$, given the available information $(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_*)$. According to the formalism of the multivariate Gaussian distributions, $(\boldsymbol{y}_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_*)$ also follows a Gaussian distribution $\mathcal{N}(\mu^*, \Sigma^*)$ whose mean and covariance matrix are

given by

$$\begin{aligned}\mu^* &= \mathcal{K}(\mathbf{X}_*, \mathbf{X})\Big(\mathcal{K}(\mathbf{X}, \mathbf{X})\Big)^{-1} \boldsymbol{y}\,, \\ \Sigma^* &= \mathcal{K}(\mathbf{X}_*, \mathbf{X}_*) \\ &\quad - \mathcal{K}(\mathbf{X}_*, \mathbf{X})\Big(\mathcal{K}(\mathbf{X}, \mathbf{X})\Big)^{-1} \mathcal{K}(\mathbf{X}, \mathbf{X}_*).\end{aligned} \tag{7}$$

Due to the random nature of this formalism, the ensemble average of all drawn $\boldsymbol{y}_*$ is reported as the most likely predicted value, i.e. μ^*.

In our analysis, we used `George` Python package, developed by Ambikasaran et al. (2015), to efficiently calculate the likelihood function as defined by Equation 5 and to make predictions based on Equation 7.

4. Particle Swarm Optimization

The Particle Swarm Optimization (PSO) technique was originally developed by Kennedy and Eberhart (1995); Shi and Eberhart (1998) to simulate the behavioral evolution of social organisms that are in group motions such as bird flocks or fish schools. This approach leverages the power of a swarm of particles to explore the parameter space hoping to find the optimum solution. In this approach, each individual particle in the parameter space is denoted by its position and velocity. The performance of each particle is evaluated based on the value of the objective function at its position. Each particle has a memory of its "best" previous position. Moreover, the best position that is ever found in the past history of the evolution of all particles is recorded. In this optimization algorithm. the position and velocity of all particles are randomly initialized and iteratively updated. In each iteration, the position and velocity of each particle is updated according to its previous "best" location and the historical "best" position of the entire group. At the end of the desired number of iterations, the optimum point is reported to be the best position ever achieved by the ensemble of particles over the entire course of their evolution.

In the following discussion, we assume that our goal is to find the minimum of an error function, $\mathcal{E}$. The number of particles, N, and the total number of iterations, M, are decided prior to calculations. In mathematical terms, the PSO algorithm is as follows: (1) In step $m = 1$, the position and velocity of N particles are randomly initialized.

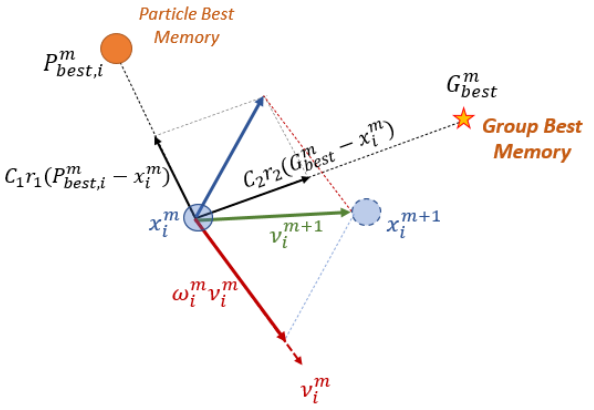

Figure 1: Evolution of the state of particle i at step m. Here, velocity v^m_i and location, x^m_i, are updated.

(2) The value of $\mathcal{E}$ is evaluated at the current position of each particle. The location where $\mathcal{E}$ takes the smallest value amongst all particles and across all processed m iterations is recorded as the best achieved minimum by the group, G^m_{best}. In addition, each individual particle memorizes its best explored location over m iterations, $P^m_{best,i}$. (3) At each step, the state of all $i = 1, \ldots, N$ particles are updated using

$$\begin{aligned} v^{m+1}_i &= \omega^m_i v^m_i + c_1 r_1 (P^m_{best,i} - x^m_i) \\ &\quad + c_2 r_2 (G^m_{best} - x^m_i), \end{aligned} \tag{8a}$$

$$x^{m+1}_i = x^m_i + v^{m+1}_i. \tag{8b}$$

Here, x^m_i and v^m_i represent the position and velocity of particle i at iteration m. The updated velocity is the superposition of three vectors: one is in the direction of the current velocity, and two are in the directions towards the best positions found by the particle and the entire group. Figure 1 schematically displays how the position of a particle (blue circle) is evolved. c_1 and c_2 are the acceleration factors. They control the dynamic of the swarm evolution with respect to the best social and personal experiences. r_1 and r_2 are taken from random uniform distributions with values ranging between 0 and 1. The semi-stochastic behavior of particles improves their odds of not dancing around the local optima. Here, ω^m_i is the inertia weight that controls the fractional impact of the particle current velocity. (4) Steps (2-3) are repeated until reaching the total number of

iterations, i.e. $m = M$. (5) The output of the algorithm is G^M_{best} where $\mathcal{E}(G^M_{best})$ is expected to be the objective minimum.

The performance of the PSO algorithm depends on how its free parameters, N, M, ω, c_1, and c_2, are chosen. It is worth mentioning that PSO does not guarantee that an optimal solution is ever found. However, not using the gradient of the objective function makes PSO favorable in the optimization of non-differentiable functions, where the classic optimization methods such as gradient descent and quasi-newton methods are not applicable.

5. The hybrid GPR-PSO methodology

Any GPR model is described by its kernel function and the values of its hyperparameters. The objective is to benefit from the power of a swarm of particles to explore the hyper-space with the aim of reaching the highest possible performance of the GPR model.

In a simple GPR framework, for a set of training data $(\boldsymbol{X}, \boldsymbol{y})$, the best hyperparameters of the kernel (Equation 4) are those that maximize the likelihood in Equation 5. The generated kernel with the optimum hyperparameters is then used in Equation 7 to evaluate the expected target values of μ^* for a set of test data points, $\boldsymbol{X}_*$. Despite the fact this approach seems to be reasonable, and in fact it is in many cases, it suffers from the over-fitting syndrome in cases where the uncertainties of the measured parameters are tiny or unknown. In the presence of such issue, the GPR model exhibits exquisite performance on the training data while it behaves poorly in the test phase.

To overcome the over-fitting problem, the available data set is divided into three different subsets. The biggest subset (around 80% of the data) is used in the training process. Two other portions are usually smaller in size and each comprises about 10% of data. One part is applied for the cross-validation purposes where the model hyperparameters are tuned and the other part is used for testing the model performance. Although in most applications all of these subsets are chosen randomly, for time series the subsets are chosen from the separated temporal regions to ensure the causal relations are preserved as much as possible. In our study, the training data is chosen from a region of time series that is preceding to the other two regions, and is followed by cross validation and test regions, respectively.

Here, to optimize the GPR kernel, we only use training and cross validation sets which are represented by $(\boldsymbol{X}, \boldsymbol{y})$ and $(\boldsymbol{X}_*, \boldsymbol{y}_*)$, respectively, and in compliance with the presented formalism earlier. The hybrid GPR-PSO method starts with randomly distributing a swarm of N particles in a hyper space defined by the free parameters of the GPR kernel, i.e. $\Theta = (\sigma_e, \sigma_f, \text{and } \ell_1, \ldots, \ell_p)$. These particle are then evolve following the procedure explained in section 4, where $\mathcal{E}(\theta)$ is the objective function that is minimized. We choose $\mathcal{E}$ to be the root mean squared difference between predictions and the real measured parameters of the cross validation data points. For a given set of hyperparameters, Θ, Equations 4 and 7 are applied consecutively to derive $\mu^* \equiv \boldsymbol{y}^{(p)}_*$, which is defined to be the prediction of the inferred GPR for the cross validation set. The error function is then defined as

$$\mathcal{E}(\Theta) = RMSE(\boldsymbol{y}_*, \boldsymbol{y}^{(p)}_*), \tag{9}$$

where $RMSE(\mathbf{A}, \mathbf{B}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(A_i - B_i)^2}$.

Ultimately, the kernel of the GPR model is constructed by accepting the best position taken by the swarm in the hyper space. In the end, we use the test set to evaluate the performance of the derived GPR model.

References

Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D.W., O'Neil, M., 2015. Fast Direct Methods for Gaussian Processes. IEEE Transactions on Pattern Analysis and Machine Intelligence 38. doi:10.1109/TPAMI.2015.2448083, arXiv:1403.6015.

Blum, M., Riedmiller, M., 2013. Electricity demand forecasting using gaussian processes, in: Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence, Citeseer.

Chen, N., Qian, Z., Nabney, I.T., Meng, X., 2014. Wind power forecasts using gaussian processes and numerical weather prediction. IEEE Transactions on Power Systems 29, 656–665. doi:10.1109/TPWRS.2013.2282366.

Gibbs, M., MacKay, D.J., 1997. Efficient Implementation of Gaussian Processes. Technical Report.

Guo, J., Chen, F., Xu, C., 2017. Traffic flow forecasting for road tunnel using pso-gpr algorithm with combined kernel function. Mathematical Problems in Engineering 2017, 2090783. URL: https://doi.org/10.1155/2017/2090783, doi:10.1155/2017/2090783.

Karbasi, M., 2017. Forecasting of multi-step ahead reference evapotranspiration using wavelet- gaussian process regression model. Water Resources Management 32. doi:10.1007/s11269-017-1853-9.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, pp. 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968.

Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning. MIT Press.

Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), pp. 69–73. doi:10.1109/ICEC.1998.699146.

Wang, P., Kim, Y., Vaci, L., Yang, H., Mihaylova, L., 2018. Short-term traffic prediction with vicinity gaussian process in the presence of missing data, in: 2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF), IEEE. pp. 1–6.

Wang, X., Liu, S., Yan, L., Wang, N., 2017/07. Energy consumption forecast based on coupling pso-gpr, in: Proceedings of the 2017 3rd International Conference on Economics, Social Science, Arts, Education and Management Engineering (ESSAEME 2017), Atlantis Press. URL: https://doi.org/10.2991/essaeme-17.2017.414, doi:https://doi.org/10.2991/essaeme-17.2017.414.