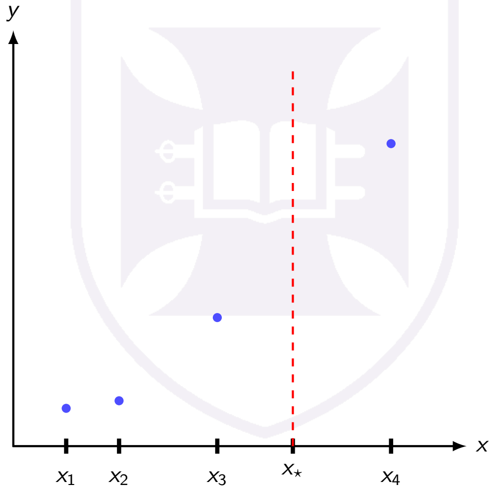
Optimizing Gaussian Processes

MSS Presentation

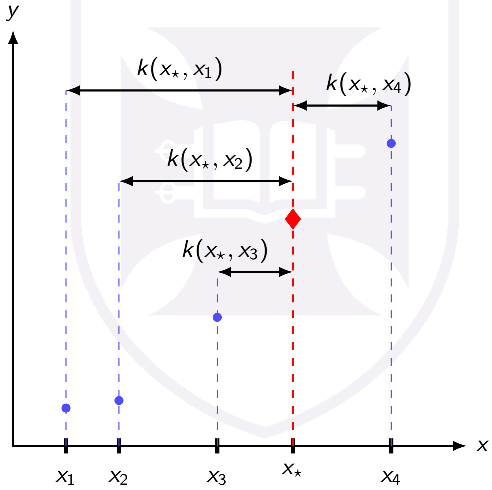
Michael Ciccotosto-Camp - 44302913

The University of Queensland
Australia

Time Series Prediction

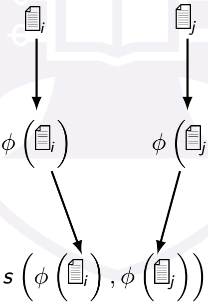


Time Series Prediction



The Kernel Trick

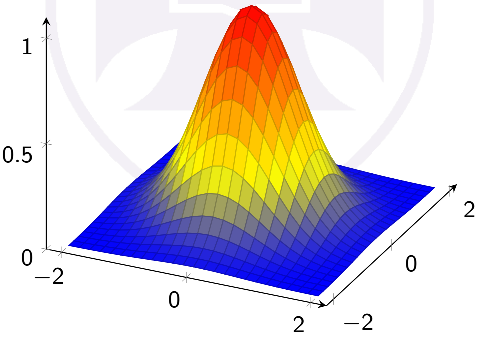
- Q: How do we get a suitable function k for computing similarity? A: Use the kernel trick!
- Suppose we have some inputs [🗎$_1, \ldots,$ 🗎$_n$] (with their corresspondding experimental observations $[y_1, \ldots, y_n]$), where 🗎$_i$ can take a number of different of form (perhaps a tree data structure or vectors of values).



- The function s provides us with some notion of similarity between inputs after they've been "transformed" into a nicer form using a *feature map* ϕ.

The Kernel Trick

- The kernel function k does all this computation is one step so that $k\left(\text{🗎}_i, \text{🗎}_j\right) = s\left(\phi\left(\text{🗎}_i\right), \phi\left(\text{🗎}_j\right)\right)$.
- Usually we have access to k, meaning we can avoid having to construct a feature map ϕ and similarity function s.
- A very common kernel function used is the RBF or Gaussian kernel.

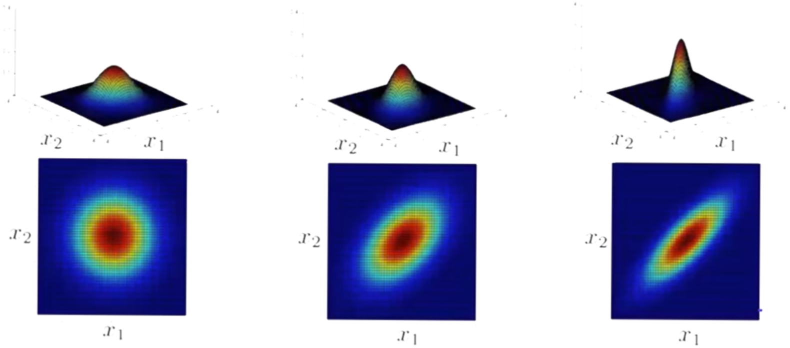


Multi-Variate Gaussian

$$\mathcal{N}\left(\begin{bmatrix}\mu_1\\ \mu_2\\ \mu_3\\ \vdots\\ \mu_n\end{bmatrix}, \begin{bmatrix}\sigma_1^2 & \sigma_{12}^2 & \sigma_{13}^2 & \cdots & \sigma_{1n}^2\\ \sigma_{21}^2 & \sigma_2^2 & \sigma_{23}^2 & \cdots & \sigma_{2n}^2\\ \sigma_{31}^2 & \sigma_{32}^2 & \sigma_3^2 & \cdots & \sigma_{3n}^2\\ \vdots & & & \ddots & \vdots\\ \sigma_{n1}^2 & \sigma_{n2}^2 & \sigma_{n3}^2 & \cdots & \sigma_{nn}^2\end{bmatrix}\right)$$

Predictions

- How do we use our data to make predictions with our kernel function?
- Within the Gaussian Process paradigm we assume that our data along with the novel point at which we would like to predict form a joint Gaussian distribution.



(Machine Learning, Stanford University, https://www.coursera.org/learn/machine-learning)

$$\mathcal{N}\left(\begin{bmatrix}\mu_1\\ \mu_2\\ \mu_3\\ \vdots\\ \mu_n\end{bmatrix}, \begin{bmatrix}\sigma_{11}^2 & \sigma_{12}^2 & \sigma_{13}^2 & \cdots & \sigma_{1n}^2\\ \sigma_{21}^2 & \sigma_{22}^2 & \sigma_{23}^2 & \cdots & \sigma_{2n}^2\\ \sigma_{31}^2 & \sigma_{32}^2 & \sigma_{33}^2 & \cdots & \sigma_{3n}^2\\ \vdots & & & \ddots & \vdots\\ \sigma_{n1}^2 & \sigma_{n2}^2 & \sigma_{n3}^2 & \cdots & \sigma_{nn}^2\end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix}\mu_1\\ \mu_2\\ \mu_3\\ \vdots\\ \mu_n\end{bmatrix}, \begin{bmatrix}\sigma_{11}^2 & \sigma_{12}^2 & \sigma_{13}^2 & \cdots & \sigma_{1n}^2\\ \sigma_{21}^2 & \sigma_{22}^2 & \sigma_{23}^2 & \cdots & \sigma_{2n}^2\\ \sigma_{31}^2 & \sigma_{32}^2 & \sigma_{33}^2 & \cdots & \sigma_{3n}^2\\ \vdots & & & \ddots & \vdots\\ \sigma_{n1}^2 & \sigma_{n2}^2 & \sigma_{n3}^2 & \cdots & \sigma_{nn}^2\end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix}y_1\\ y_2\\ \vdots\\ y_n\\ ?\end{bmatrix}, \begin{bmatrix}k(\boldsymbol{x}_1,\boldsymbol{x}_1) & k(\boldsymbol{x}_1,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1,\boldsymbol{x}_n) & k(\boldsymbol{x}_1,\boldsymbol{x}_*)\\ k(\boldsymbol{x}_2,\boldsymbol{x}_1) & k(\boldsymbol{x}_2,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2,\boldsymbol{x}_n) & k(\boldsymbol{x}_2,\boldsymbol{x}_*)\\ \vdots & & \ddots & & \vdots\\ k(\boldsymbol{x}_n,\boldsymbol{x}_1) & k(\boldsymbol{x}_n,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_n,\boldsymbol{x}_n) & k(\boldsymbol{x}_n,\boldsymbol{x}_*)\\ k(\boldsymbol{x}_*,\boldsymbol{x}_1) & k(\boldsymbol{x}_*,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_*,\boldsymbol{x}_n) & k(\boldsymbol{x}_*,\boldsymbol{x}_*)\end{bmatrix}\right)$$

- How do we compute that missing mean within our MVN from the previous slide?

- How do we compute that missing mean within our MVN from the previous slide?
- We can use the following theorem: (Marginals and conditionals of an MVN), suppose $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]$ is jointly Gaussian with parameters

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

then the posterior conditional is given by

$$\begin{aligned} \boldsymbol{x}_2 \mid \boldsymbol{x}_1 &\sim \mathcal{N}\left(\boldsymbol{x}_2 \mid \boldsymbol{\mu}_{2|1}, \boldsymbol{\Sigma}_{2|1}\right) \\ \boldsymbol{\mu}_{2|1} &= \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\left(\boldsymbol{x}_1 - \boldsymbol{\mu}_1\right) \\ \boldsymbol{\Sigma}_{2|1} &= \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}. \end{aligned}$$

$$\mathcal{N}\left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_* \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) & k(\boldsymbol{x}_1, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_n) & k(\boldsymbol{x}_2, \boldsymbol{x}_*) \\ \vdots & & \ddots & & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & k(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) & k(\boldsymbol{x}_n, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_*, \boldsymbol{x}_1) & k(\boldsymbol{x}_*, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_*, \boldsymbol{x}_n) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_* \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) & k(\boldsymbol{x}_1, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_n) & k(\boldsymbol{x}_2, \boldsymbol{x}_*) \\ \vdots & & \ddots & & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & k(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) & k(\boldsymbol{x}_n, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_*, \boldsymbol{x}_1) & k(\boldsymbol{x}_*, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_*, \boldsymbol{x}_n) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_\star \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} & \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}}^\intercal \\ \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}} & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_* \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) & k(\boldsymbol{x}_1, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_n) & k(\boldsymbol{x}_2, \boldsymbol{x}_*) \\ \vdots & & \ddots & & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & k(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) & k(\boldsymbol{x}_n, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_*, \boldsymbol{x}_1) & k(\boldsymbol{x}_*, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_*, \boldsymbol{x}_n) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_\star \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{XX}} & \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}}^\intercal \\ \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}} & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

- The mean and covariance can then be computed using the theorem from before as

$$\begin{aligned} y_\star &= \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}} \boldsymbol{K}_{\boldsymbol{XX}}^{-1} \boldsymbol{y} \\ \mathbb{V}\left[y_\star\right] &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}} \boldsymbol{K}_{\boldsymbol{XX}}^{-1} \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}}^\intercal. \end{aligned}$$

$$\mathcal{N}\left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_* \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) & k(\boldsymbol{x}_1, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_n) & k(\boldsymbol{x}_2, \boldsymbol{x}_*) \\ \vdots & & \ddots & & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & k(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) & k(\boldsymbol{x}_n, \boldsymbol{x}_*) \\ k(\boldsymbol{x}_*, \boldsymbol{x}_1) & k(\boldsymbol{x}_*, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_*, \boldsymbol{x}_n) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

$$\mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_\star \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} & \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}}^\intercal \\ \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}} & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

- The mean and covariance can then be computed using the theorem from before as

$$y_\star = \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-1} \boldsymbol{y}$$

$$\mathbb{V}\left[y_\star\right] = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-1} \boldsymbol{K}_{\boldsymbol{X}_\star \boldsymbol{X}}^\intercal .$$

- Another way of looking at the prediction is seeing it as a linear combination of n kernel evaluations centered at the input $\boldsymbol{x}_\star$

$$y_\star = \sum_{i=1}^{n} \alpha_i k\left(\boldsymbol{x}_i, \boldsymbol{x}_\star\right)$$

Unoptimized GPR

Algorithm 1: Unoptimized GPR

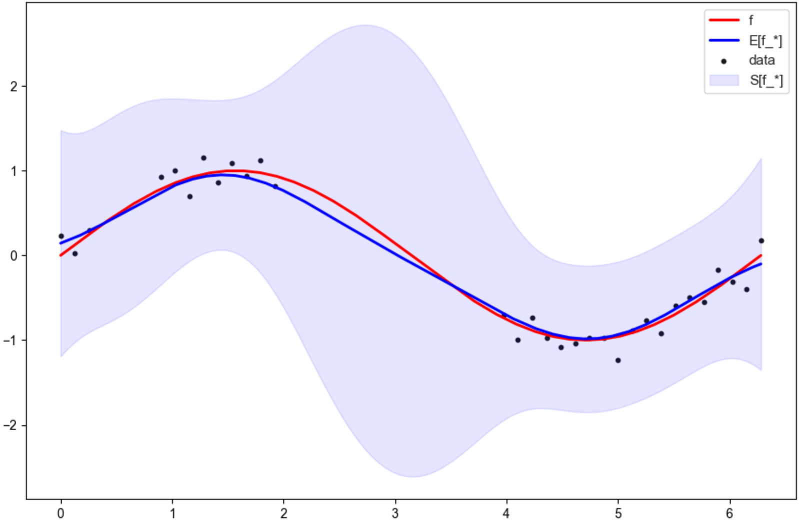
input : Observations $\boldsymbol{X}, \boldsymbol{y}$ and a test input $\boldsymbol{x}_{\star}$.
output: A prediction $y_{\star}$ with its corresponding variance $\mathbb{V}\left[y_{\star}\right]$.

$\boldsymbol{L} = \text{cholesky}\left(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}\right)$
$\alpha = \text{lin-solve}\left(\boldsymbol{L}^{\mathsf{T}}, \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{y}\right)\right)$
$y_{\star} = \boldsymbol{K}_{\boldsymbol{x}_{\star}\boldsymbol{X}}\alpha$
$\boldsymbol{v} = \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{K}_{\boldsymbol{x}_{\star}\boldsymbol{X}}\right)$
$\mathbb{V}\left[y_{\star}\right] = k(\boldsymbol{x}_{*}, \boldsymbol{x}_{*}) - \boldsymbol{v}^{\mathsf{T}}\boldsymbol{v}$
return $y_{\star}, \mathbb{V}\left[y_{\star}\right]$

Implementation

```python
def gp_reg_pred(X_train, Y_train, x_pred, sigma):
    n, d = X_train.shape
    # Create the Gram matrix corresponding to the training data set.
    K = exact_kernel(X_train, sigma=sigma)
    # Noise variance of labels.
    s = np.var(Y_train.squeeze())
    L = np.linalg.cholesky(K + s*np.eye(n))
    # Compute the mean at our test points.
    Lk = np.linalg.solve(L, exact_kernel(X_train, x_pred, sigma=sigma))
    Ef = np.dot(Lk.T, np.linalg.solve(L, Y_train))
    # Compute the variance at our test points.
    K_ = exact_kernel(x_pred, sigma=sigma)
    Vf = np.diag(K_) - np.sum(Lk**2, axis=0)
    return Ef, Vf
```

sin Function Prediction with Added Noise



Stock Market Prediction