Gaussian processes. Theory and applications in predictive modeling of spatiotemporal phenomena

Martin Andreev Asenov
s1247380

Report
Autonomous Systems Research
School of Informatics
University of Edinburgh

2016

Abstract

In this report we present a tutorial on Gaussian Processes. The report explains the theory behind them, provide implementation details and conduct series of experiments to give a further intuition of how they work. We touch on number of more advance topics like connections with neural nets and hierarchical Gaussian Processes. In the second part of the report we look into the problems of active sensing and modeling spatiotemporal phenomena and why Gaussian Processes are well suited for those tasks. Accompanying to the report the reader is also encourage to check a presentation and code base, which can be found on `https://github.com/masenov/GP_Intro`

Table of Contents

Chapter 1

Introduction

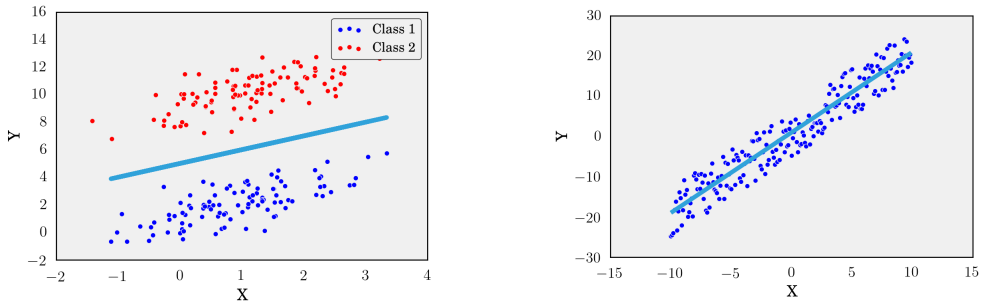
Machine learning can be broadly divided into three different categories - supervised learning, unsupervised learning and reinforcement learning. Supervised learning deals with the problem of trying to make a prediction based on input information. Supervised learning can be further divided into classification and regression. When we want to predict a finite number of types or classes we have a classification problem. Examples of this include recognition of different categories of images and the language a text is written in. Regression deals with the problem when the prediction we want to make is a continuous variable. For example we can try to predict a certain stock price or a price of property in London. Both of the problem involve coming up with non-linear, high dimensional curve that we either fit through the data (regression) or draw decision boundary (classification).

Unsupervised learning deals with unlabeled data. We try to infer structure of the data we observe, identifying groups of similar examples. Methods include clustering and density estimation. Finally there is reinforcement learning. Reinforcement learning deals with the problem of finding a suitable actions in different situations in order to maximize a reward or achieve a goal.

In this report we will focus on solving regression problems using Gaussian Processes. In the simplest form those problems can be defined in terms of the pairs

$$\{x_n, y_n\}_{n=1}^{N} \tag{1.1}$$

where x_n and y_n are vectors. We try to predict y_n based on x_n. Through an extension we can model classification problems similarly. Gaussian Processes are also used in reinforcement learning. However the latter two topics are outside of the scope of this report.



(a) (b)

Figure 1.1: **Classification and regression.** (a) In classification problems we try to come up with a decision boundary to discriminate between different classes. (b) In regression we try to fit a curve through our data.

Chapter 2

Gaussian distribution

2.1 Definition

Gaussian (or normal) distribution is probably the most used distribution, because of it's good properties as discussed later. The univariate Gaussian (having only 1 random variable) is defined in eq.2.1.

$$N(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

The univariate Gaussian has two hyperparameters the mean μ and standard deviation σ. The Gaussian has a "bell shape" curve as seen in fig.2.1. The effect hyperparameters have on the shape of the distribution can be seen in fig.2.2.

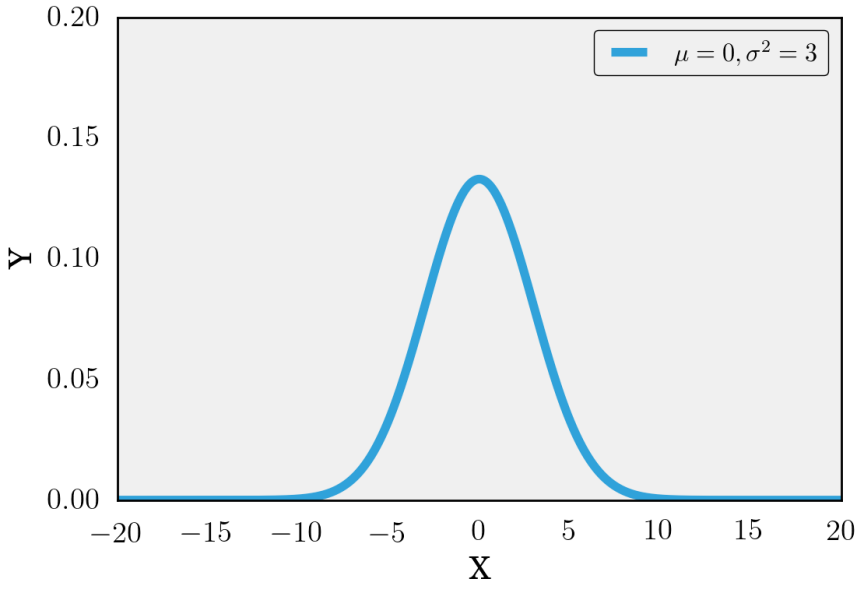


Figure 2.1: **Gaussian distribution.**

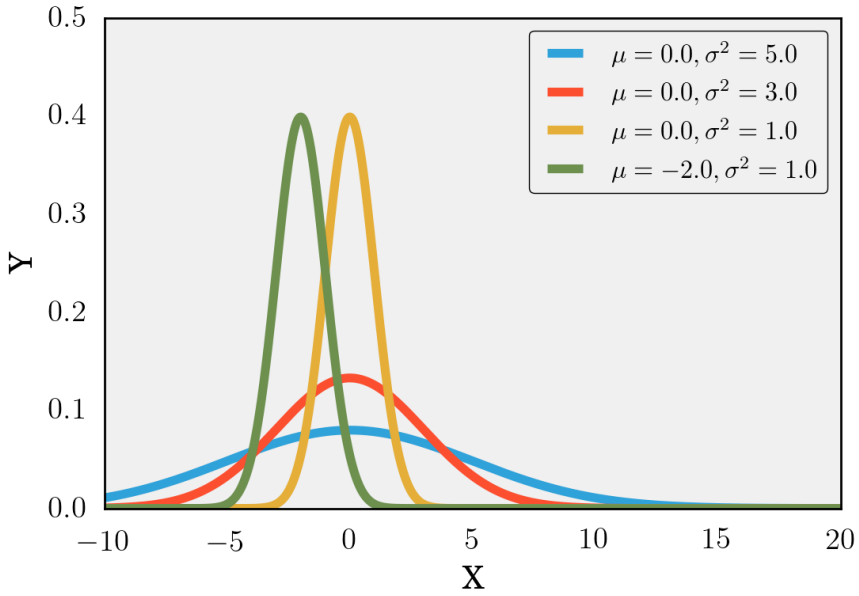


Figure 2.2: **Effect on μ and σ on the Gaussian.**

By changing μ we translate the distribution along the X axis and change the most likely value of the distribution. On the other hand σ controls the shape of the distribution - whether we have a tall and narrow bell shaped curve, or wide and short. This specifies the range of values we get - when we have smaller variance, we only get values close to the mean. As the variance increases we get wider range of values around the mean. The univariate Gaussian distribution can be extended to multivariate by defining as in eq.2.2.

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{2.2}$$

Here, similarly to the univariate Gaussian, μ is a mean vector, and Σ is the covariance matrix. The mean vector simply consists of the means of the different dimensions of the distribution. The covariance matrix is defined as in eq.2.3.

$$\Sigma_{ij} = cov(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] \tag{2.3}$$

An example $2D$ Gaussian can be seen in fig.2.3.

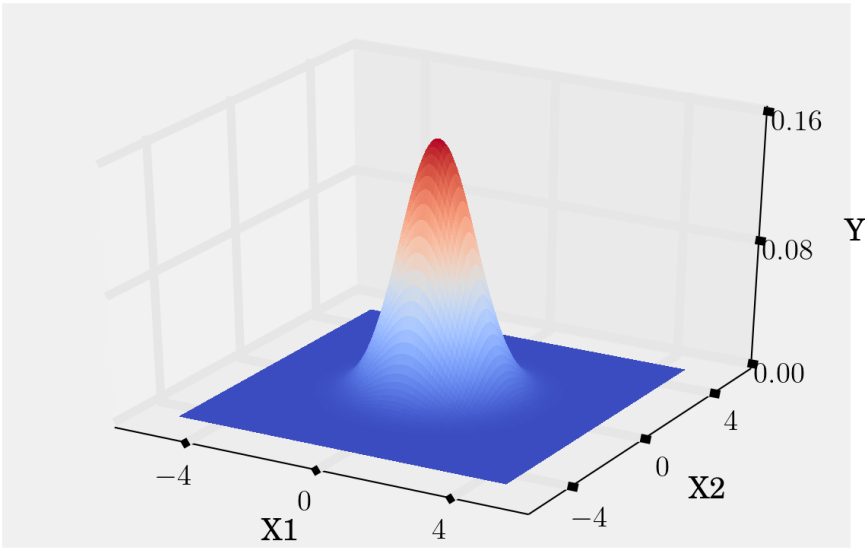


Figure 2.3: **2D Gaussian distribution.** We have defined $\mu = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

A more convenient way to visualize a $2D$ Gaussian can be by viewing it from the top, referred as a contour plot. An example of the same Gaussian, but as a contour plot can be seen in fig.2.4.

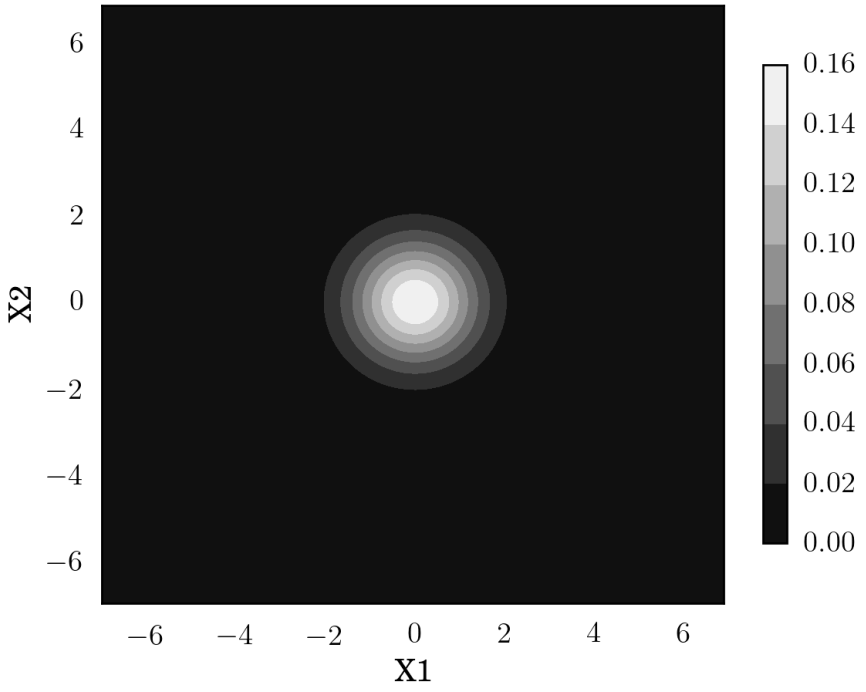


Figure 2.4: **2D Gaussian distribution, contour plot.** Alternative visualization of fig.2.3

In order to get a better intuition of how the covariance matrix Σ and the mean vector μ change the $2D$ normal distribution we can vary them and plot the resulting contour plots. The mean again changes position of the mass of the distribution as seen in fig.2.5. The covariance matrix specifies how correlated the values $X1$ and $X2$ are as seen in fig.2.6.

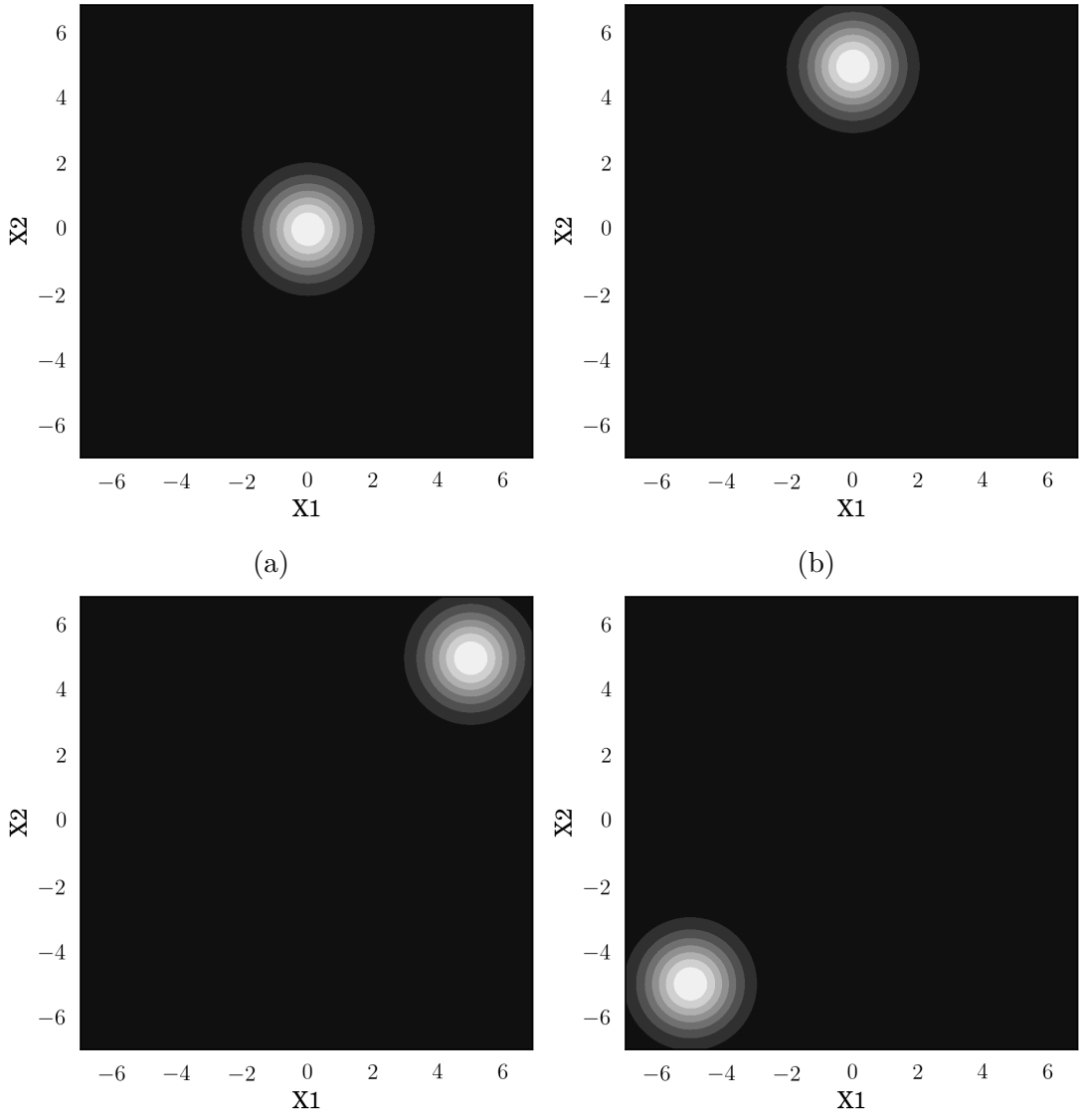
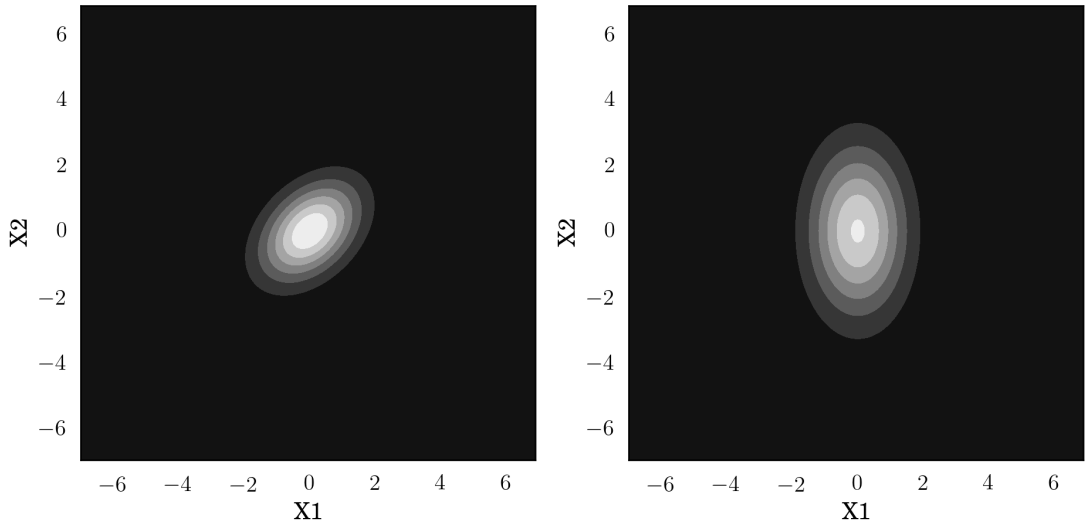
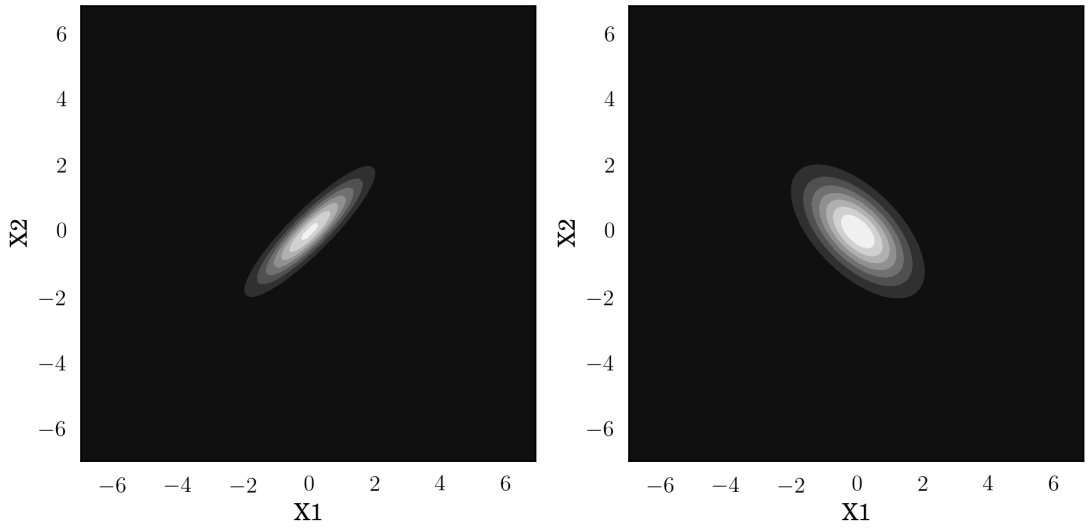


Figure 2.5: **2D Gaussian distribution for different value of μ.** (a) $\mu = [0\ \ 0]$ (b) $\mu = [5\ \ 0]$ (c) $\mu = [5\ \ 5]$ (d) $\mu = [-5\ \ -5]$



(a)

(b)



(c)

(d)

Figure 2.6: **2D Gaussian distribution for different value of Σ.** (a) $\Sigma = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$ (b) $\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$ (c) $\Sigma = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$ (d) $\Sigma = \begin{bmatrix} 1 & -0.6 \\ -0.6 & 1 \end{bmatrix}$

We can make some interesting observation by "fixing" the first dimension of a Gaussian and observing the other one. Formalizing this we want to compute the probability $P(X_2|X_1)$. This probability is in fact a univariate Gaussian, which is one of the properties that makes the normal distribution so useful.

2.2 Sampling

Given a specified Gaussian distribution we can draw samples from it. In fig.2.7 we show how we can draw samples using rejection sampling in the univariate

case. We can bound the Gaussian with a box, where the y coordinates run from 0 to value of the Gaussian evaluated at its mean point. For the x coordinates we can use the interval $[-3\sigma, 3\sigma]$, which captures 99.7% of the values. We can sample uniformly in those two defined intervals generating points $[x_i, y_i]$. If they are below the distribution we accept the samples, if not we reject them. We can extend this method and draw samples from higher dimensional Gaussians. In fig.2.8 we can see random samples from a $2D$ Gaussian. We see from fig.2.7 that we end up rejecting a lot of the samples. There are other more efficient ways to sample from a normal distribution [4][16][6]. We present this method just for completeness, as it gives an intuition of how one might go about implementing sampling.

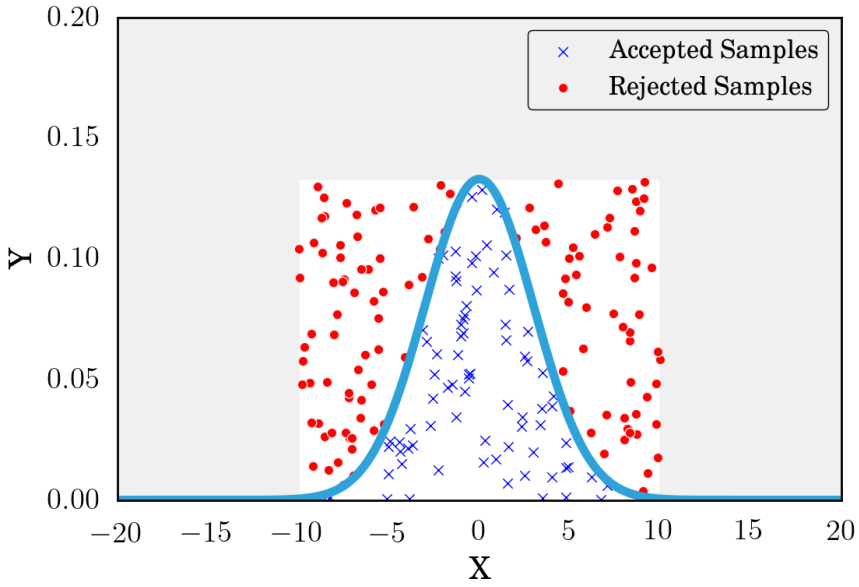


Figure 2.7: **Drawing samples from 1D Gaussian using rejection sampling.**

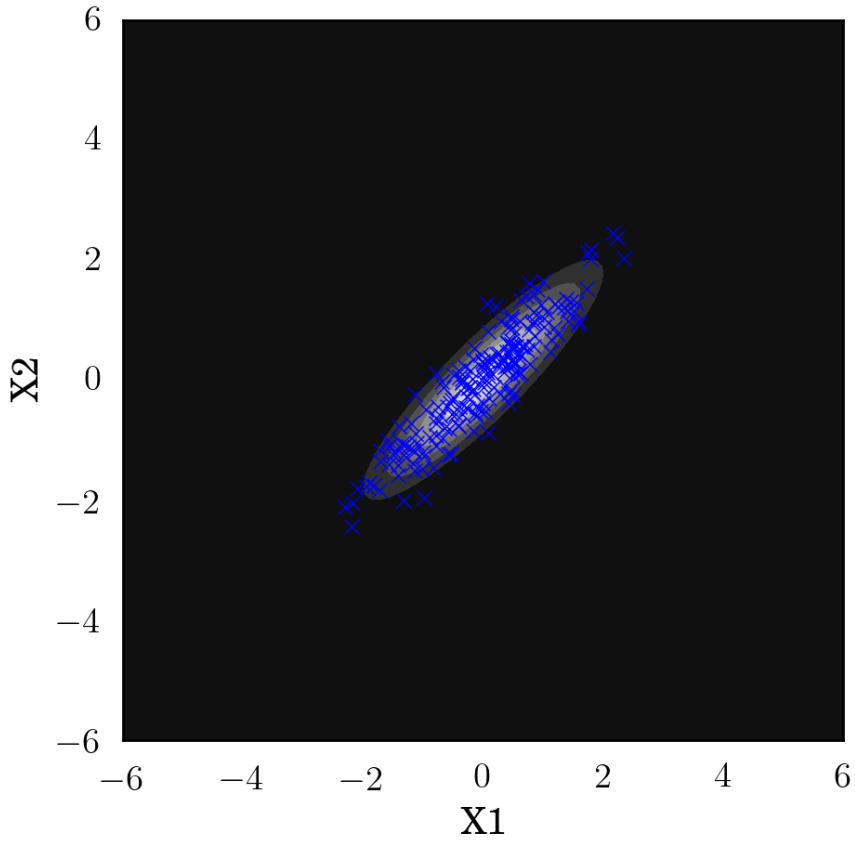


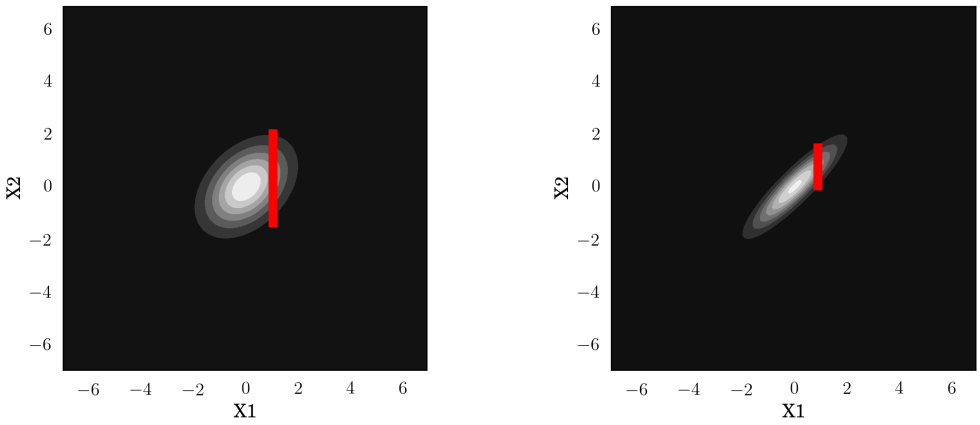
Figure 2.8: **Drawing samples from 2D Gaussian.**

2.3 Error bars

A useful metric from statistics that we will use is standard error. We define it in eq.2.4, where σ is the standard deviation of our random variable and N is the number of samples that we draw.

$$\sigma_M = \frac{\sigma}{\sqrt{N}} \tag{2.4}$$

With standard error we can quantify uncertainty of a random variable. From the definition we see that intuitively the smaller the variance of the random variable and the more samples we draw, the smaller the error thus the less the uncertainty. In fig.2.9 we show the error bars for the conditional distribution $P(X_2|X_1)$. When $N = 1$ we define the standard error as one standard error which is just the standard deviation of the distribution.



(a) (b)

Figure 2.9: **Error bars in** $2D$ **Gaussian.** We fix $X1 = 1$ and draw samples from $X2$. We show the error bars for $N = 1$ sample and different covariance matrices (a) $\Sigma = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$ (a) $\Sigma = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}$

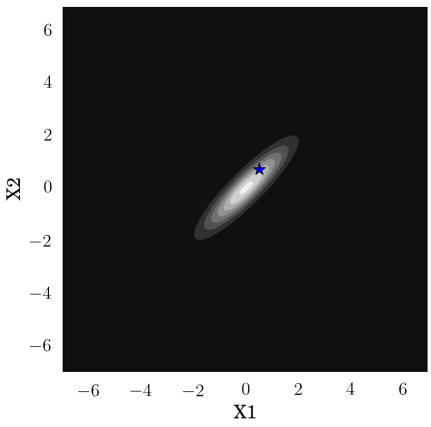
Chapter 3

Gaussian Processes

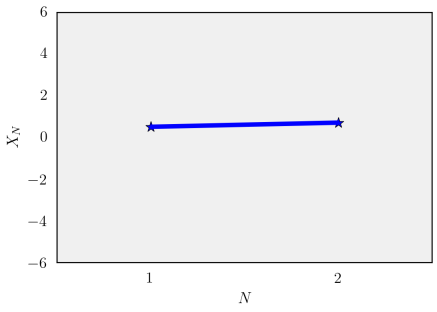
In this chapter we build up on our knowledge about Gaussian distributions in order to define Gaussian Processes.

3.1 Alternative visualization of samples of multivariate Gaussian

In this section we present an alternative way of visualizing samples from a multivariate Gaussian distribution. We will plot the different components of X for each sample we have. In fig.3.1 we again give an example with a $2D$ Gaussian. Instead of visualizing the samples on a contour plot, we can have the indexes of the vector X along the x axis, namely 1 and 2, and the corresponding elements of the vector on the y axis, X_1 and X_2.



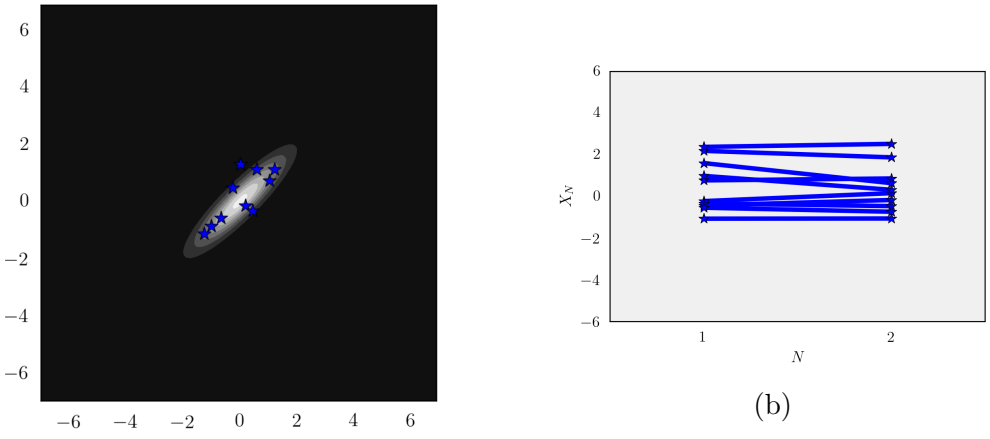
(a)



(b)

Figure 3.1: **Alternative way of plotting a Gaussian.** (a) Plotting $2D$ Gaussian as a contour plot, where every sample is a point on the contour plot. (b) Plotting only the one sample we have drawn from the Gaussian.

In fig.3.2 we show more samples from the same distribution.



(a) (b)

Figure 3.2: **10 samples from a** $2D$ **Gaussian.**

1

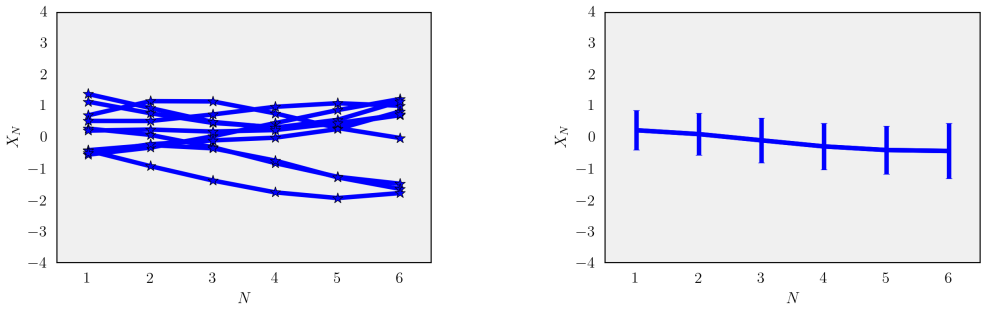
Having defined this way of visualizing samples from a multivariate normal distribution, we can now visualize samples from a distribution with more than three dimensions. In fact we can extend this to any dimension we want. For example let's have the $6D$ Gaussian distribution with mean

$$\mu = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1}$$

and covariance matrix

$$\Sigma = \begin{bmatrix} 1 & 0.95 & 0.8 & 0.6 & 0.41 & 0.25 \\ 0.95 & 1 & 0.95 & 0.8 & 0.6 & 0.41 \\ 0.8 & 0.95 & 1 & 0.95 & 0.8 & 0.6 \\ 0.6 & 0.8 & 0.95 & 1 & 0.95 & 0.8 \\ 0.41 & 0.6 & 0.8 & 0.95 & 1 & 0.95 \\ 0.25 & 0.41 & 0.6 & 0.8 & 0.95 & 1 \end{bmatrix} \tag{3.2}$$

We can again draw samples as in fig.3.3, (a). Measuring the standard deviation at each point, we can calculate the error bars as discussed earlier.



(a) (b)

Figure 3.3: **6D Gaussian.** (a) Ten samples from the Gaussian. (b) Error bars showing the standard deviation at each dimension.

Looking at the plot, we make two observations. We notice that the lines we generate start to look like a nonlinear regression. They also appear to be smooth. This suggests that there is correlation between points close to each other. We also observe the structure of the covariance matrix we draw the samples from. The answers to these questions comes from the way we generate the covariance matrix, using a specific kernel.

3.2 Kernel function

First we will give a definition of a kernel function. A kernel function specifies how to construct a covariance matrix for a multivariate normal distribution. In fact the covariance matrix from eq.3.2 was not random, but generated from a specific kernel. The kernel that we use is 'squared exponential' - a decreasing function of the distance between the two points as seen in eq.3.3.

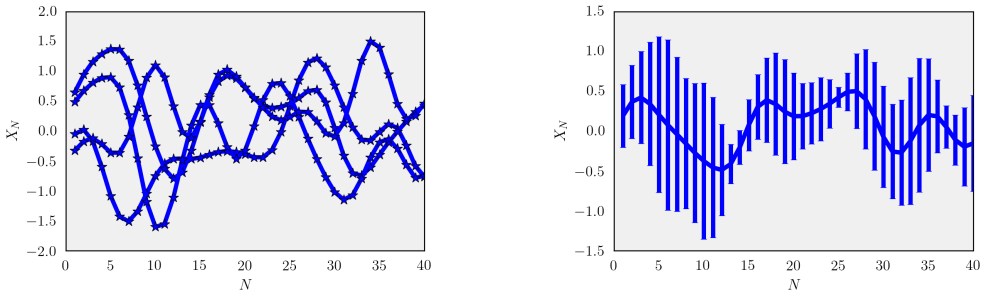
$$k(x_n, x_{n'}) = \sigma_f^2 exp\left(-\frac{1}{2l^2}(x_n - x_{n'})^2\right) \tag{3.3}$$

Here x_n is the index of the particular dimensional of our Gaussian, in our case 1 through 6. σ_f is the vertical lengthscale, l is the horizontal lengthscale. Using a kernel, we can generate a covariance matrix from eq.3.4.

$$cov(y_n, y_{n'}) = k(x_n, x_{n'}) + \sigma_v^2 \delta_{nn'} \tag{3.4}$$

y_n and $y_{n'}$ specify row and column of the element of the covariance matrix, we are currently generating the value for. σ_v specifies the noise in our predictions and $\delta_{nn'}$ is the direct delta function.

This framework allows us to generate a covariance matrix with any size. We can also extend our mean vector as much as we like, since so far we've defined it as a zero vector. In fig.3.4 we show 4 samples drawn from $40D$ Gaussian.



(a) (b)

Figure 3.4: **Samples from 40D Gaussian using 'squared exponential' kernel function.** $\sigma_v = 0$, $l = 1$, $\sigma_f = 1$. (a) Four samples from the Gaussian. (b) Error bars showing the standard deviation at each dimension.

In sec.3.4 we go in more details on the effect the hyperparameters have on the sample functions we generate.

3.3 Definition

In this section we can finally formally define a Gaussian Processes ($\mathcal{GP}$).

Definition 3.3.1 *A Gaussian processes is a collection of random variables with the property that the joint distribution of any finite subset is a Gaussian.*

Definition 3.3.2 *A Gaussian processes is fully specified by a mean and a covariance function.*

In order to give a better intuition about $\mathcal{GP}$ we examine fig.3.5. So far we've been thinking about the visualizations of samples of multivariate normal distribution as in fig.3.5, (a). We will rename the axis $N \to X$ and $X_N \to Y$ as in fig.3.5, (b) and think about each sample from the multivariate normal distribution as function that maps $x \to y$. So far we were setting length of the X axis with the dimension of multivariate normal distribution we draw samples from. However, the only place where we use the different X_i is in eq.3.3 and eq.3.4. So far we were using only integer numbers for the different dimensions, to calculate the correlations between them. However we can "relabel" the different dimensions with any float number we want and calculate the covariance matrix correspondingly. Thus we see in fig.3.5, (c) we draw the samples from a $40D$ Gaussian, but represent X only in a interval with length 6 - from 1 to 7. So we can set an arbitrary interval for the X axis. In other words the dimension of the Gaussian from which we draw samples, only controls the precision with which we want to represent a function. Moreover we don't have to take all the X_i at the same distance from one another, as seen in fig.3.5, (d). Finally, just for clarity, we will just show the different functions and use different colors as seen in fig.3.5, (e) and (f).

Thus a $\mathcal{GP}$ is a framework for sampling functions with certain properties given mean and covariance function as seen in eq.3.5.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{3.5}$$

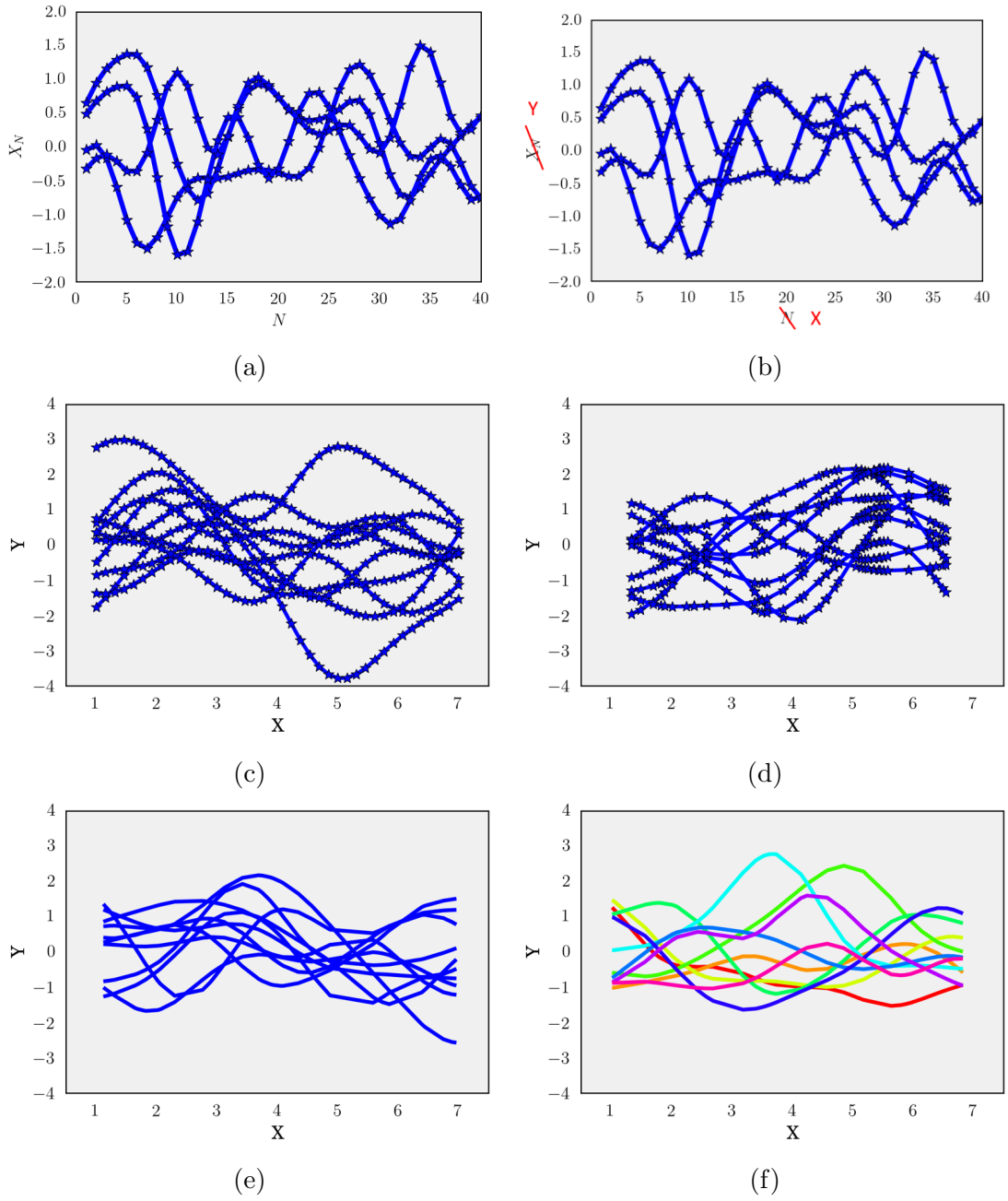


Figure 3.5: **Defining a $\mathcal{GP}$ as a method for sampling functions.** (a) Plotting a samples from a high dimensional Gaussian as defined in fig.3.1 (b) Renaming the two axis (c) We can have a arbitrary interval for X axis (d) Points taken from this function does not have to be equally spaced (e) Showing just the sampled functions (f) The sampled functions with different colors

3.4 Kernel hyperparameters

In this section we explore the effect of hyperparameters of the squared exponential kernel defined in eq.3.3 and eq.3.4. The kernel function has three hyperparameters - noise σ_v, vertical lengthscale σ_f and horizontal lengthscale l. Their effects can be seen in fig.3.6, 3.7 and 3.8. The noise hyperparameter σ_v allows us to model noise in our data. The vertical lengthscale σ_f is a scaling factor determining the interval Y of our generated functions. The horizontal lengthscale l controls how smooth the generated functions are. The hyperparameters are set based on our knowledge and assumptions before we observe any data.

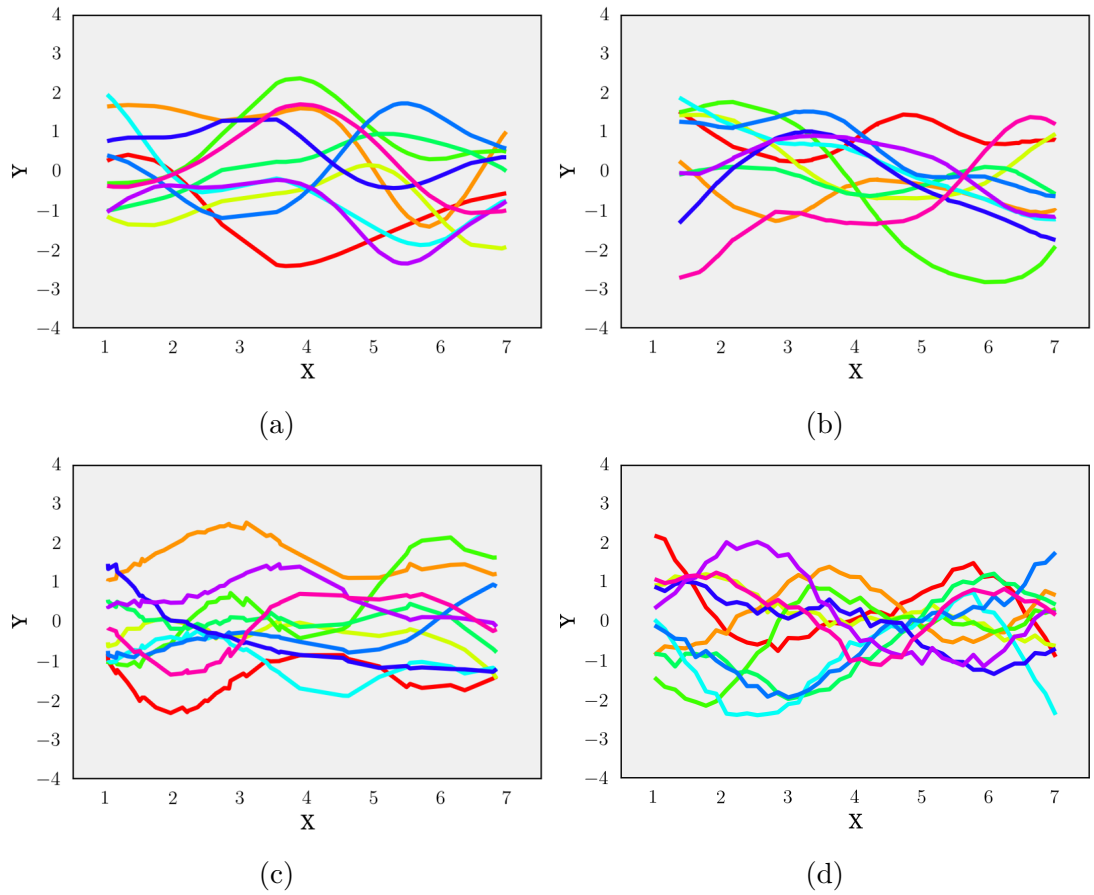


Figure 3.6: **Effect of noise hyperparameter σ_v.** We fix $\sigma_f = 1$ and $l = 1$ and vary σ_v. (a) $\sigma_v = 0$ (b) $\sigma_v = 0.01$ (c) $\sigma_v = 0.05$ (d) $\sigma_v = 0.1$

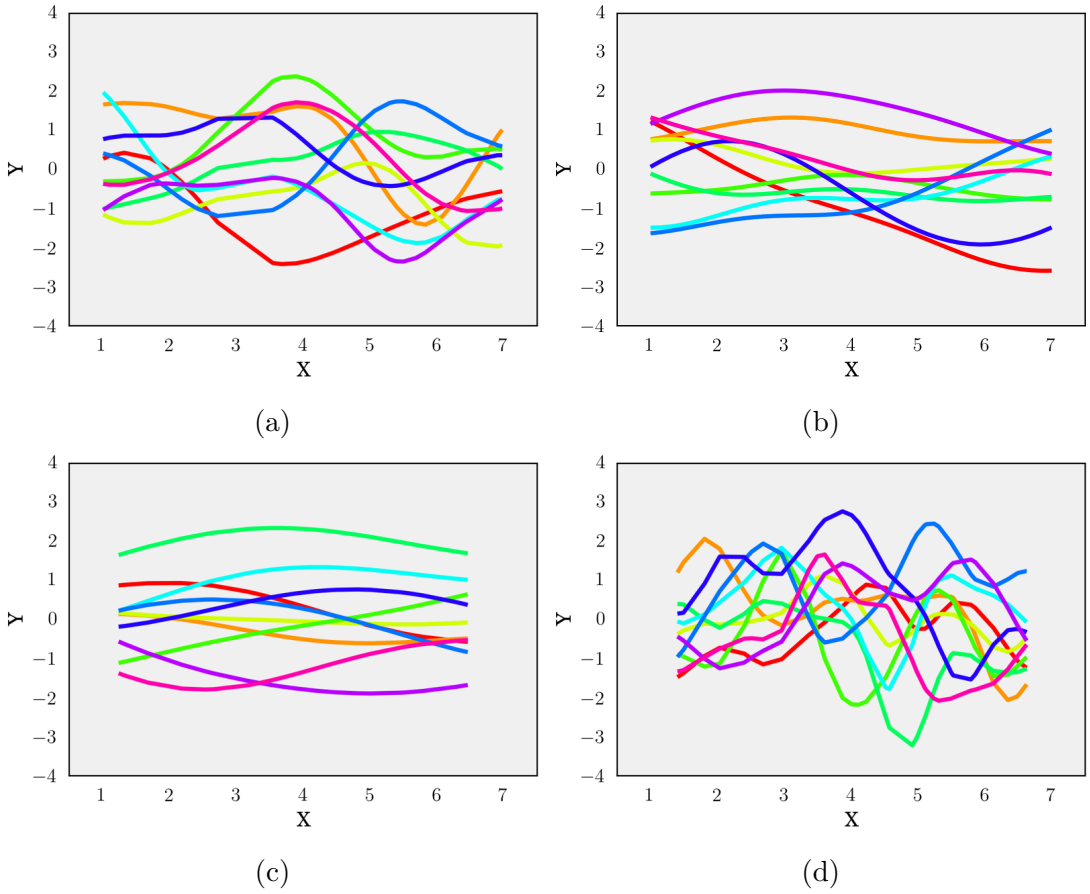


Figure 3.7: **Effect of noise hyperparameter l.** We fix $\sigma_v = 0$ and $\sigma_v = 1$ and vary l. (a) $l = 1$ (b) $l = 3$ (c) $l = 5$ (d) $l = 0.5$

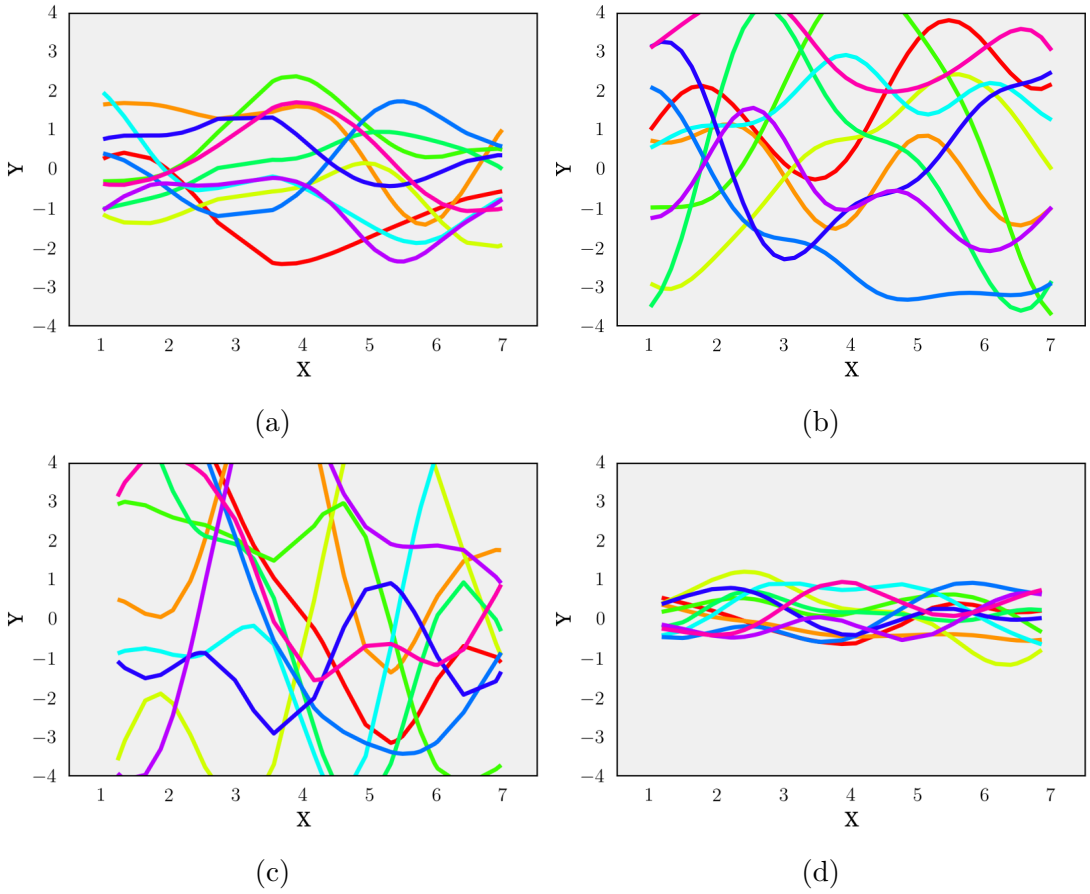


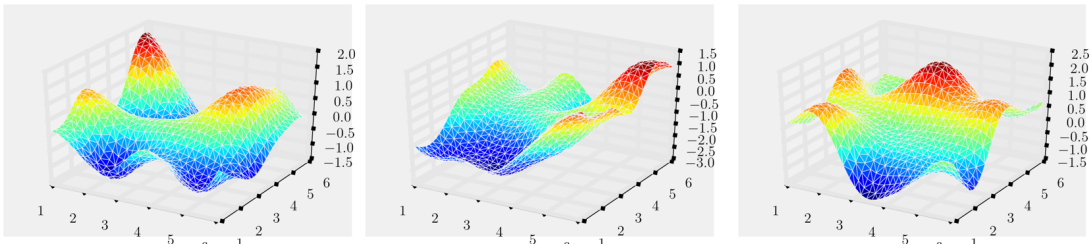
Figure 3.8: **Effect of noise hyperparameter σ_f.** We fix $\sigma_v = 0$ and $l = 1$ and vary σ_f. (a) $\sigma_f = 1$ (b) $\sigma_f = 2$ (c) $\sigma_f = 3$ (d) $\sigma_f = 0.5$

3.5 Two-dimensional input space

Thinking about $\mathcal{GP}$ as a way of sampling functions as defined in eq.3.5, we can extend this beyond the linear case. As seen in eq.3.6 and eq.3.7, the only modification we have to do to our kernel function is modify the way we calculate the correlation between the different dimensions of the Gaussian to include more dimensions. We can intuitively see how this works in fig.3.9. Here the samples from our $\mathcal{GP}$ are smooth surfaces similar to the smooth functions we were generting in sec.3.4.

$$cov(y_n, y_{n'}) = k(x_n, x_{n'}) + \sigma_v^2 \delta_{nn'} \tag{3.6}$$

$$k(x_n, x_{n'}) = \sigma_f^2 exp\left(-\sum_{d=1}^{D} \frac{1}{2l^2}(x_{dn} - x_{dn'})^2 \right) \tag{3.7}$$



(a) (b) (c)

Figure 3.9: **Sample 2D functions (surfaces) generated from $\mathcal{GP}$ with squared exponential kernel**

Chapter 4

Gaussian Processes for nonlinear regression

In previous chapters we defined $\mathcal{GP}$ and how they can used to generate functions with certain properties. In this chapter we introduce data and investigate how we can use $\mathcal{GP}$ in order to model the data in the context of nonlinear regression.

4.1 Bayesian Inference

Bayes' theorem is one of the fundamental theorems in probability theory. The theorem has the form as in eq.4.1.

$$\pi(\theta|x) = \frac{f(x|\theta)p(\theta)}{f(x)} \tag{4.1}$$

The terms of the theorem as usually referred as in eq.4.2.

$$Posterior = \frac{Likelihood * Prior}{Evidence} \tag{4.2}$$

The term $f(x)^{-1}$ does not depend on θ, it is just a normalization constant. Thus we can calculate the posterior distribution up to proportionality. The form of the Bayes' Theorem, as in eq.4.3, is the basis for Bayesian Inference.

$$\pi(\theta|x) \propto f(x|\theta)p(\theta) \tag{4.3}$$

An important concept in Bayesian inference is conjugate priors. Conjugate prior allows us to calculate the posterior in closed-form. Gaussian distribution has the nice property that it is self-conjugate - if we have Gaussian prior and likelihood, this ensures that our posterior will also be a Gaussian.

Definition 4.1.1 *If the posterior distribution $\pi(\theta|x)$ has the same form as the prior distribution $p(\theta)$, then the prior $p(\theta)$ is called conjugate prior for the likelihood function $f(x|\theta)$.*

We define our prior distribution $p(\theta)$ as:

$$p(\theta) = \frac{1}{\sqrt{2\pi}\sigma_f} e^{-\frac{(x-\mu_f)^2}{2\sigma_f^2}} \tag{4.4}$$

and likelihood $f(x|\theta)$ as:

$$f(x|\theta) = \frac{1}{\sqrt{2\pi}\sigma_g} e^{-\frac{(x-\mu_g)^2}{2\sigma_g^2}} \tag{4.5}$$

Then the posterior distribution is again drawn from a normal distribution, with the form $\pi(\theta|x) \sim N(\mu_{fg}, \sigma_{fg})$, where

$$\sigma_{fg} = \sqrt{\frac{\sigma_f^2 \sigma_g^2}{\sigma_f^2 + \sigma_g^2}} \tag{4.6}$$

and

$$\mu_{fg} = \frac{\mu_f \sigma_g^2 + \mu_g \sigma_f^2}{\sigma_f^2 + \sigma_g^2} \tag{4.7}$$

Full derivation can be found in [14]. In fig.4.1 we show Bayesian inference with three data points.

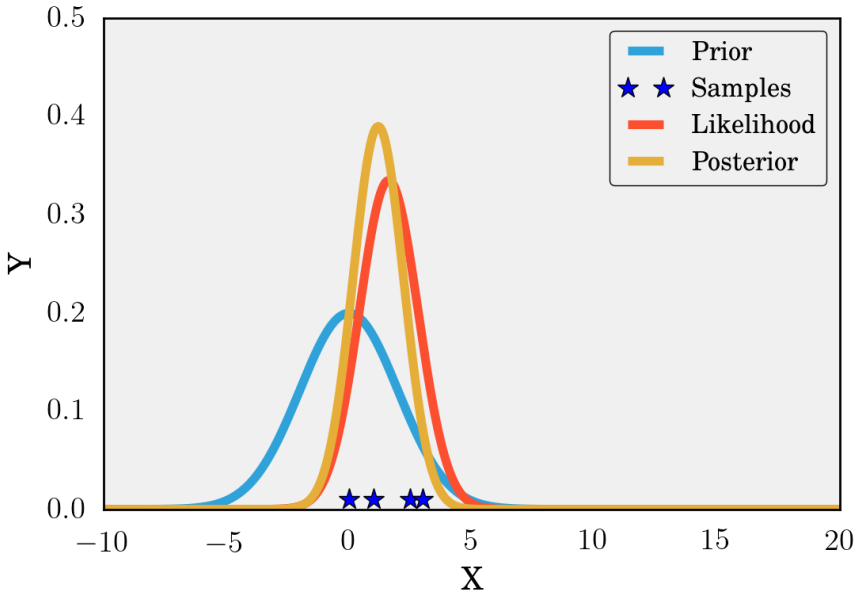


Figure 4.1: **Bayesian inference.**

4.2 Inference in Gaussian Processes

In the previous section we defined what Bayesian inference is and how we can apply it using Gaussian distributions. In this section we describe how we can use $\mathcal{GP}$ to do inference from data. In eq.3.5 we defined $\mathcal{GP}$ as a method for sampling functions. Each of these functions is described by a set of points, which we get from taking one sample from a high dimensional Gaussian distribution. The coordinates of the different dimensions of the sample define the function. We can represent our prior functions before we observe any data as in eq.4.8.

$$f_* \sim N(0, \Sigma(X_*, X_*)) \tag{4.8}$$

Now we introduce a set of data points $\{x_n, y_n\}_{n=1}^{N}$. In a vector form we can represent them as $\{X, f\}$. We can use X_* from our prior and X from our data to calculate covariance matrices using the kernel from eq.3.4. Next we can define the joint Gaussian distribution as in eq.4.9.

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} \Sigma(X, X) & \Sigma(X, X_*) \\ \Sigma(X_*, X) & \Sigma(X_*, X_*) \end{bmatrix}\right) \tag{4.9}$$

Finally, we can calculate the conditional distribution of functions, given our prior and data in eq.4.10. Since we are conditioning two Gaussian distributions, we know that the posterior will also be a Gaussian distribution.

$$f_* | X_*, X, f \sim N(\mu_{f_*}, \Sigma_{f_*}) \tag{4.10}$$

Calculating the mean vector μ_{f_*} and covariance matrix Σ_{f_*} is derived in great details in [2] (sec. 2.3.1). We can see the results in eq.4.11, 4.12.

$$\mu_{f_*} = \Sigma(X_*, X)\Sigma(X, X)^{-1} f \tag{4.11}$$

$$\Sigma_{f_*} = \Sigma(X_*, X_*) - \Sigma(X_*, X)\Sigma(X, X)^{-1}\Sigma(X, X_*) \tag{4.12}$$

Finally we can implement a simple example of nonlinear regression with $\mathcal{GP}$ as in fig.4.2.

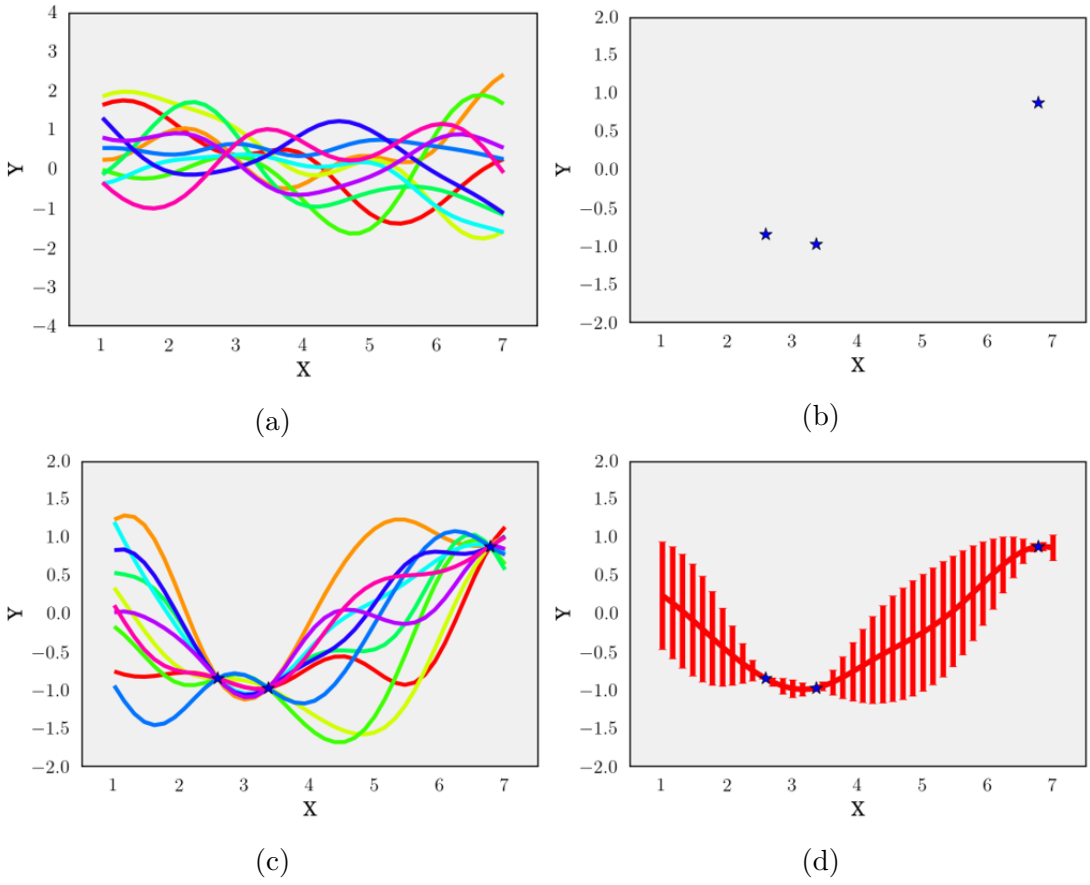


Figure 4.2: **Inference in $\mathcal{GP}$.** (a) Prior (b) Data (c) Posterior (d) Error bars

4.3 Benefits and comparison with other methods

Doing regression with $\mathcal{GP}$ has a couple of useful properties. First of all unlike neural nets for example, $\mathcal{GP}$ are a non-parametric model. We don't have to do apply many iteration of a learning algorithm like backpropagation. Learning a model consists of only setting the hyperparameters and calculating a conditional probability distribution of two Gaussians. Moreover $\mathcal{GP}$ provide a framework for quantifying uncertainty. This proves to be of great importance in different fields where we can only make limited number of noisy measurements. In this report we discuss only squared exponential kernel. However there are a wide variety of other kernels, which can generate and model different types of functions. In fact we can also combine different functions to compose a kernel, the only requirement being that the kernel has to generate positive definite covariance matrix [17]. We can use $\mathcal{GP}$ for classification similarly to neural network by applying sigmoid nonlinearity [12]. An interesting theoretical result is that $\mathcal{GP}$ can simulate shallow neural networks with one hidden layer with infinite number of neurons [19]. This combines with the fact that we can represent any function using a shallow neural

network, suggests that $\mathcal{GP}$ are indeed a powerful model[3]. They've been some recent advances to scale $\mathcal{GP}$ similar to neural networks by introducing hierarchical structure to improve learning [5].

Chapter 5

Predictive modeling of spatiotemporal phonomena

In this chapter we examine the problems of active sensing and modeling spatiotemporal phenomena and how they can be solved using $\mathcal{GP}$. We start by giving definitions and motivation for the topics, continue by developing a model for the ocean temperature in the Pacific Ocean and finish the chapter by discussing other applications and possible extensions.

5.1 Definitions and review

First we define spatiotemporal phenomena and active sensing. The two problems are often studied together, as the complexity of modeling spatiotemporal phenomena and inability to have full observations over it, leads to necessity of using active sensing.

Definition 5.1.1 *Spatiotemporal phenomena is an event relating to, or existing in both space and time.*

Definition 5.1.2 *Active sensing, or active information gathering, is collecting the most useful information about a problem and then using the gathered information to do inference with the goal of maximizing the accuracy of the inference while minimizing the quantity of information gathered.*

There have been a significant work in the field. Efficient monitoring of spatiotemporal phenomena and their dynamics is discussed in [18]. The phenomena discussed is water quality monitoring in rivers and lakes. The goal specified is to maximize the information collected, while taking into account the limitation of the sensor devices and robots they are using. Hence Gaussian Processes are a good choice since they can quantify the amount of information they have collected. Moreover they can pick locations they want to go next, based on the places with higher uncertainty. The paper suggests possible extensions for path planning for multiple robots, instead of a single one.

In modeling spatiotemporal a common problem seems to be few incorrect extreme measurements leading to inaccurate model [11]. $\mathcal{GP}$ tend to a reproduced field around a those few extreme measurements, while predictions being low, in desirable location. However by using log-measurements, mitigates the issue as it removes extremity and remove skewness.

Another key idea in conducting inference with $\mathcal{GP}$ is lazy evaluation [7]. We continuously pick a place with high uncertainty - conduct measurements - make inference and update our model - pick a place with high uncertainty - etc. Usually correlation decreases exponentially as the distance between points increases. However often variables apart from each other are dependent. By exploiting locality in kernel functions we can achieve significant speed up of the algorithm [7]. $\mathcal{GP}$ also prove to be effective in modeling spatiotemporal phenomena over long period of time [9].

5.2 Modeling ocean temperature using Gaussian Processes

In this section we conduct an experiment to demostrate $\mathcal{GP}$ in practice. We use El Nino Data Set [10]. The dataset consists of 178 080 meteorological measurements across the Pacific - air temperature, relative humidity, surface winds, sea surface temperature, etc. The measurements were made from 25 places across the Pacific Ocean over the span of 5 years. For the purpose of the report, we will try to model the air temperature for a fixed location (156 longitude, -6 latitude) as in fig.5.1.

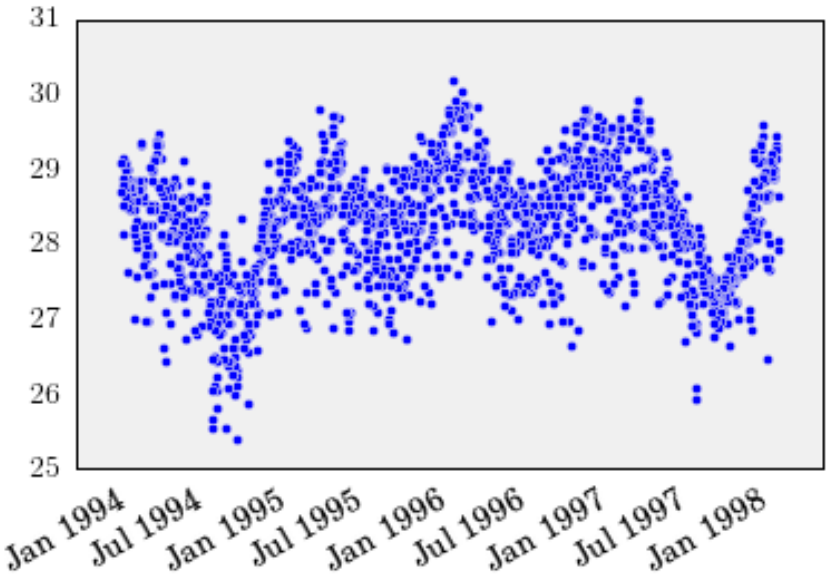


Figure 5.1: **Air temperature for 156 longitude, -6 latitude (Pacific Ocean) from Jan 1994 til Jan 1998 [10].**

We can set some priors from information outside of the data source we use. In this problem we set the mean function μ to be constant 27. We set the noise $\sigma_v = 0.1$, the horizontal lengthscale $l = 100$ days, vertical lengthscale $sigma_f = 1$ and use $40D$ Gaussian to represent our functions. We also take limited number of measurements. From the 1480 single measurements we have we take only 30 and conduct inference with them. The results can be seen in fig.5.2. We can already see that even though we take only 30 points we have already have a reasonable fit for the data.

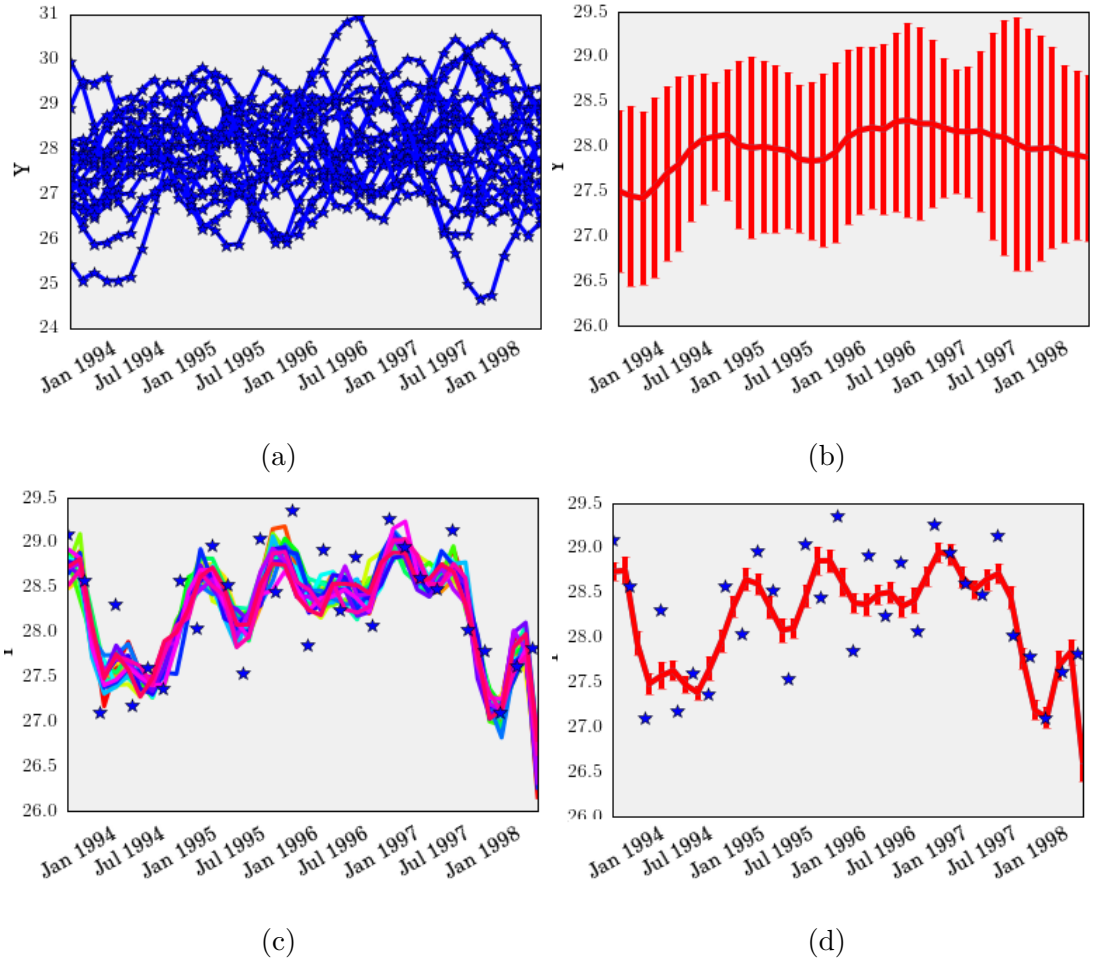


Figure 5.2: $\mathcal{GP}$ **for modeling some of the data from the El Nino Data Set [10].** Taking 30 equally spaced data points, $\sigma_v = 0.1$, $l = 100$, $sigma_f = 1$, samples from a $40D$ Gaussian (a) Samples from the prior (b) Error bars for the prior (c) Samples from the posterior (d) Error bars bars for the posterior

Next we increase to include 100 measurements and increase the dimension of the Gaussian we sample our functions from to $200D$. The results can be seen in fig.5.3. Finally just for clarity we can plot the error bars we generate on the actual data as in fig.5.4.

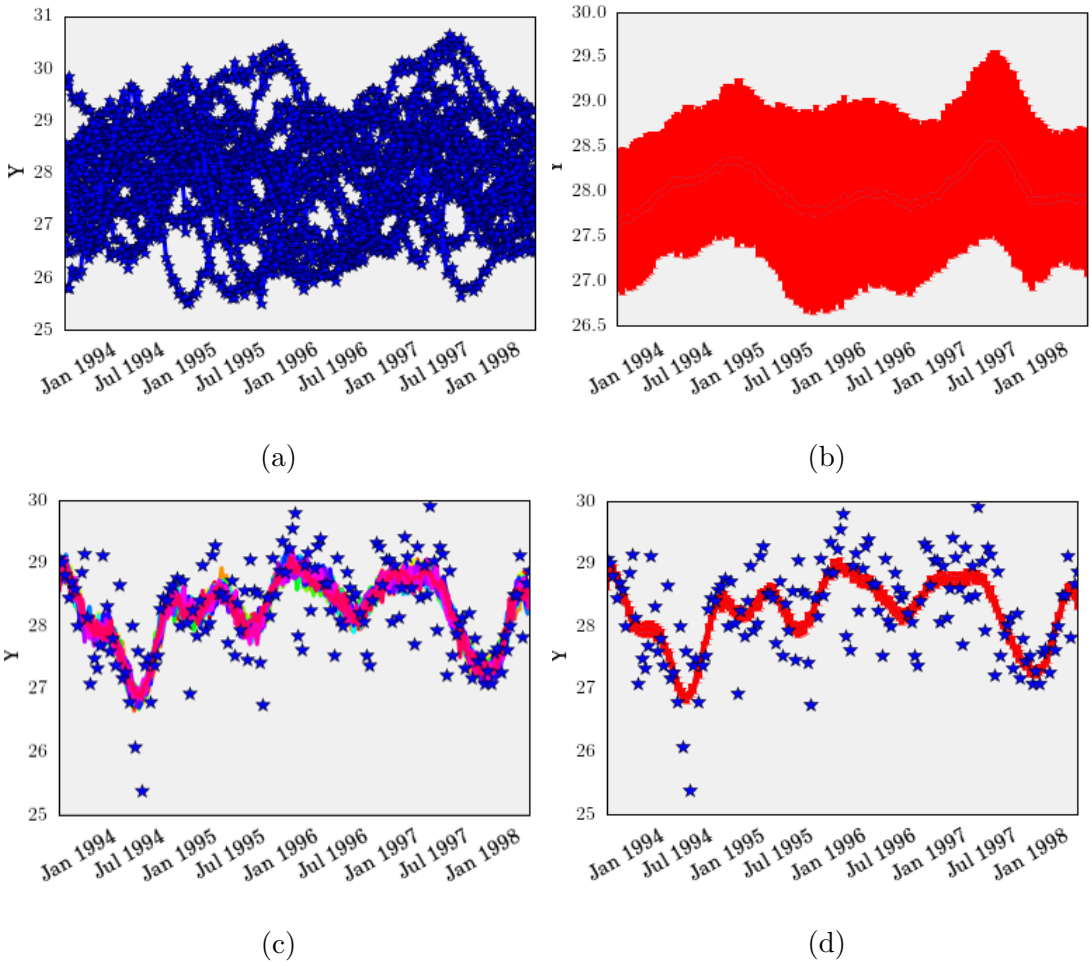


Figure 5.3: $\mathcal{GP}$ **for modeling some of the data from the El Nino Data Set [10].** Taking 100 equally spaced data points, $\sigma_v = 0.1$, $l = 100$, $sigma_f = 1$, samples from a $200D$ Gaussian (a) Samples from the prior (b) Error bars for the prior (c) Samples from the posterior (d) Error bars bars for the posterior

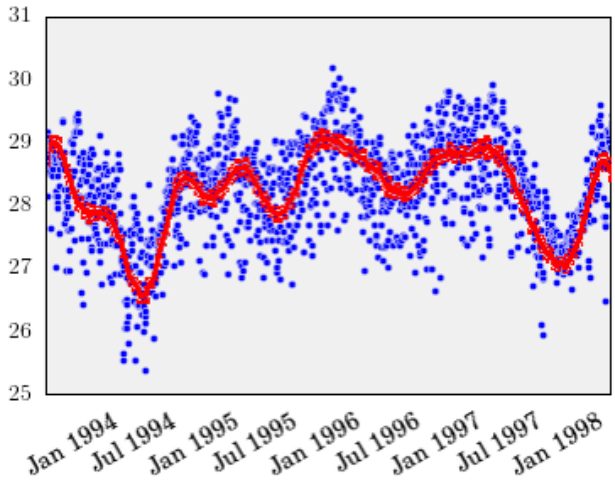
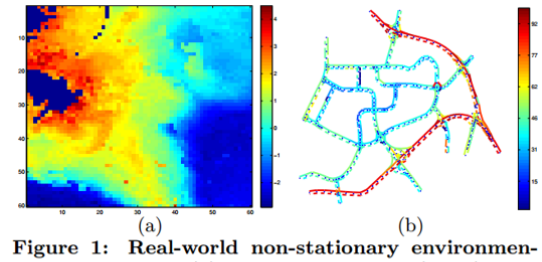


Figure 5.4: **Final results.**

5.3 Other Applications

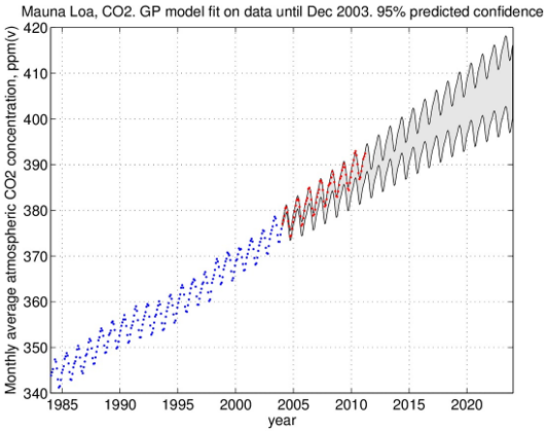
We already presented some applications in sec.5.1 while discussing useful properties and extensions of $\mathcal{GP}$. Other successful examples include modeling plankton densities and road traffic. An interesting example is the prediction of CO2 concentration in [17]. Measurements continued to be collected 5 years after the book was published and the model described them very well [8]. Other applications include balancing of objects [13] and information gathering for classification [1].



(a) (b)

Figure 1: Real-world non-stationary environmental phenomena: (a) Plankton density (chl-a) phenomenon (measured in mg/m^3) in log-scale in Gulf of Mexico, and (b) traffic (road speeds) phenomenon (measured in km/h) over an urban road network.

(a)



(b)

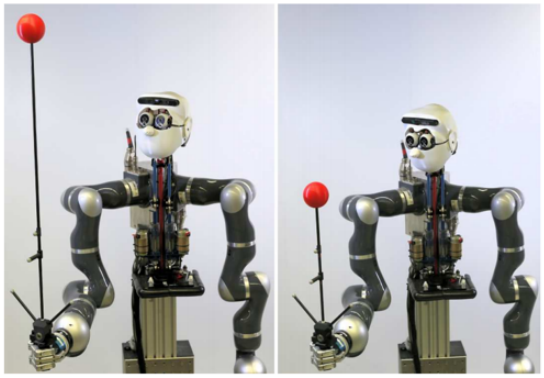
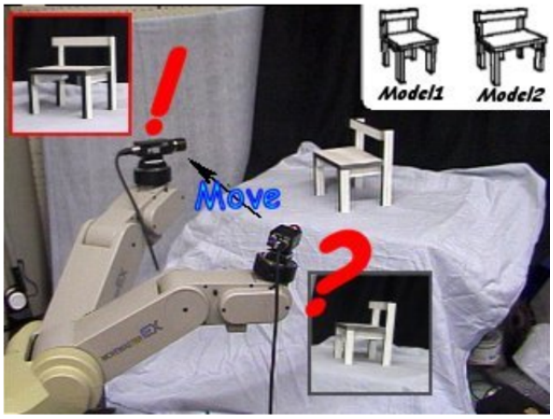


FIGURE 1.1: Robot Apollo balancing two inverted poles. These experimental platforms are used as a demonstrators of the automatic tuning framework.

(c)



(d)

Figure 5.5: **$\mathcal{GP}$ in different applications.** (a) Modeling plankton density and urban road network [15] (b) CO2 concentration [17] and [8] (c) Balancing an inverted poles from a Apollo robot [13] (d) Decision making [1]

Chapter 6

Summary

In this report we presented some of the fundamental theory that make Gaussian Processes ($\mathcal{GP}$) a powerful machine learning tool. We focus on using $\mathcal{GP}$ for nonlinear regression and discuss possible applications in classification and reinforcement learning. We briefly discuss some interesting theory and connections between $\mathcal{GP}$ and neural nets. Finally we examine the problem of active sensing and predictive modeling of spatiotemporal phenomena. The report implements an example model of nonlinear regression for the air temperature above the Pacific Ocean. The code base for this report as well as accompanying presentation can be found on `https://github.com/masenov/GP_Intro`

Bibliography

[1] Nikolay Asenov Atanasov. Active information acquisition with mobile robots. 2015.

[2] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[3] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[4] Siddhartha Chib and Bradley P Carlin. On mcmc sampling in hierarchical longitudinal models. *Statistics and Computing*, 9(1):17–26, 1999.

[5] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Practical learning of deep gaussian processes via random fourier features. *arXiv preprint arXiv:1610.04386*, 2016.

[6] Walter R Gilks and Pascal Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, pages 337–348, 1992.

[7] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM, 2005.

[8] RF Keeling, SC Piper, AF Bollenbacher, and SJ Walker. Atmospheric co2 values (ppmv) derived from in situ air samples collected at mauna loa, hawaii, usa. *Scripps Institution of Oceanography (SIO), University of California, La Jolla, California USA*, 2009.

[9] Robert E Kopp, Andrew C Kemp, Klaus Bittermann, Benjamin P Horton, Jeffrey P Donnelly, W Roland Gehrels, Carling C Hay, Jerry X Mitrovica, Eric D Morrow, and Stefan Rahmstorf. Temperature-driven global sea-level variability in the common era. *Proceedings of the National Academy of Sciences*, page 201517056, 2016.

[10] M. Lichman. UCI machine learning repository, 2013.

[11] Kian Hsiang Low, John M Dolan, and Pradeep Khosla. Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. *arXiv preprint arXiv:1305.6129*, 2013.

[12] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[13] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic lqr tuning based on gaussian process optimization: Early experimental results. In *Second Machine Learning in Planning and Control of Robot Motion Workshop at International Conference on Intelligent Robots and Systems*, 2015.

[14] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

[15] Ruofei Ouyang, Kian Hsiang Low, Jie Chen, and Patrick Jaillet. Multi-robot active sensing of non-stationary gaussian process-based environmental phenomena. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 573–580. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[16] Albert Parker and Colin Fox. Sampling gaussian distributions in krylov spaces with conjugate gradients. *SIAM Journal on Scientific Computing*, 34(3):B312–B334, 2012.

[17] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[18] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34:707–755, 2009.

[19] Christopher KI Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.