Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

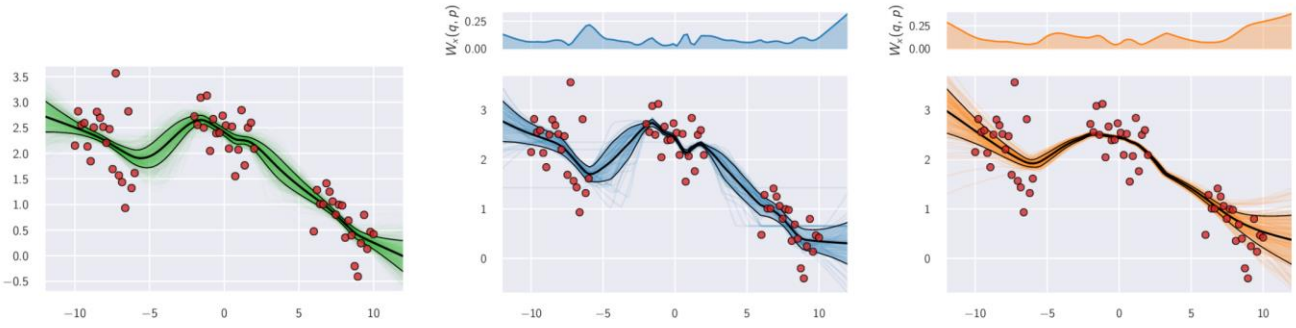
Gaussian Process

Daniel Dold
HTWG Konstanz
Institute for Optical Systems

GEFÖRDERT VOM

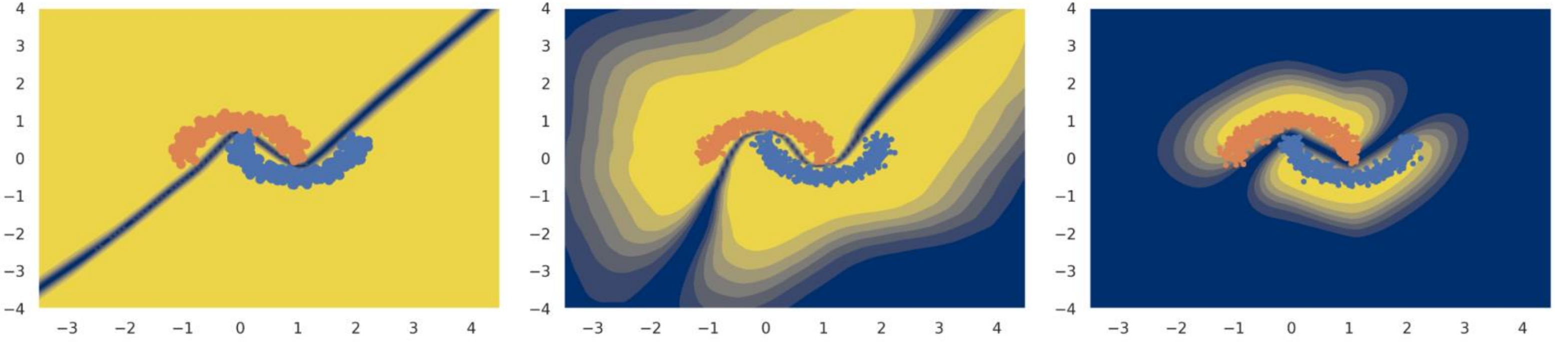
Bundesministerium
für Bildung
und Forschung

Motivation



(a) Exact (b) Deep Ensembles (c) Variational Inference

Figure from *[1]



(a) ResNet + Softmax (b) FFN + DKL (c) DUE

Figure from *[2]

*[1]: Wilson, A. G., & Izmailov, P. (2020). Bayesian Deep Learning and a Probabilistic Perspective of Generalization.

*[2]: van Amersfoort, J., Smith, L., Jesson, A., Key, O., & Gal, Y. (2021). Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression

Motivation

Its all about correlations

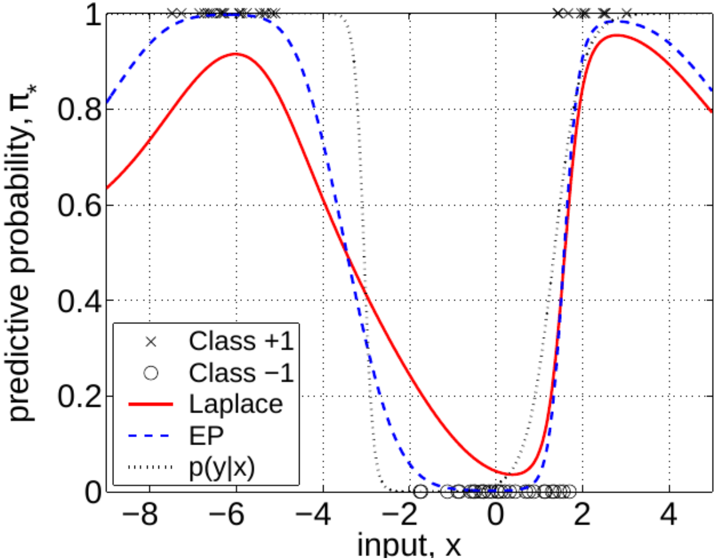
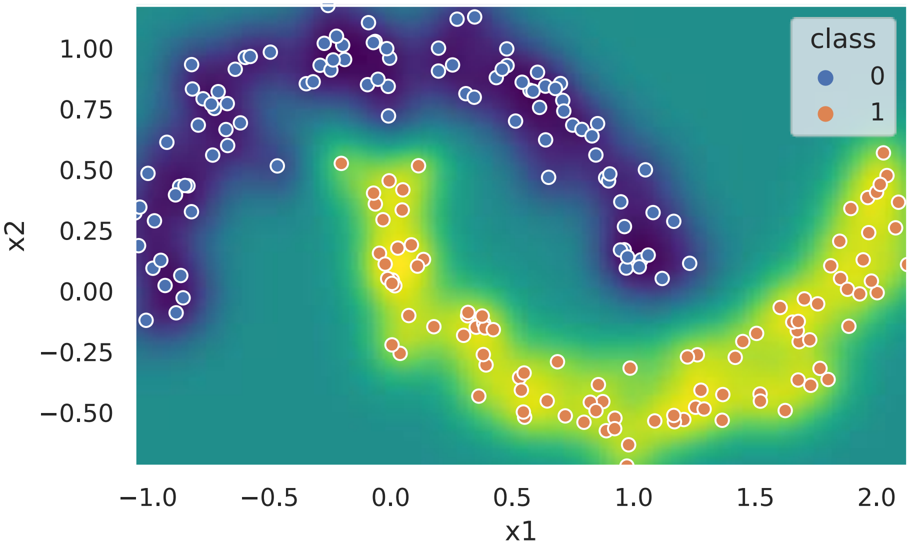


Figure from *[1]

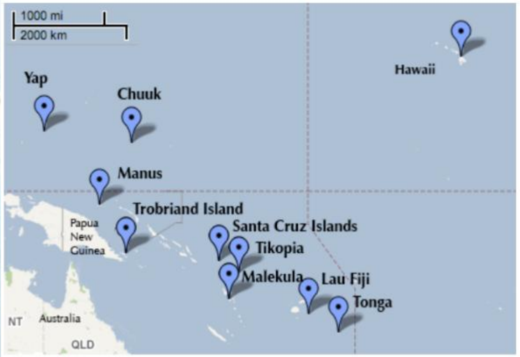


*[1]: C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006

Number of Tools on Islands

- Total number of tools T_i of island i
- Simple Model
  - $T_i \sim Poisson(\lambda_i)$
  - $\lambda_i = \alpha P_i^{\beta}$
  - P is log population

Neglects Spatial Autocorrelation / Neighboring Islands do trade



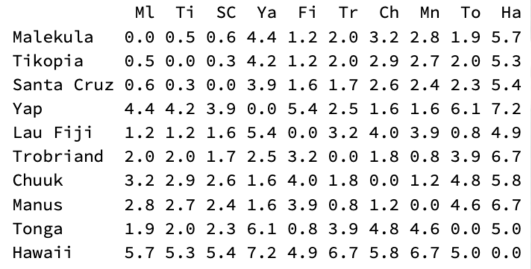
Images and example from: McElreath, R. (2018). Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC.

Islands – Taking the spatial correlation into account

Number of Tools on Islands

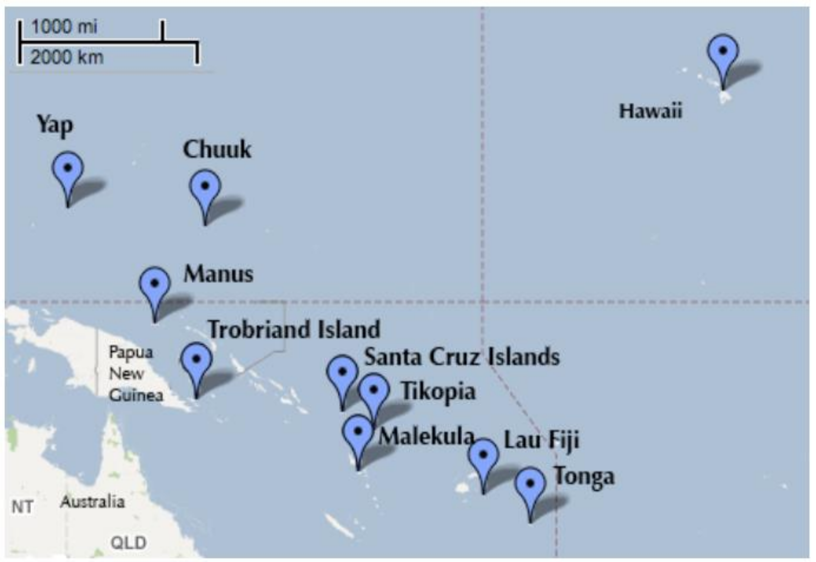
- Total number of tools T_i of island i
- Simple Model
  - $T_i \sim Poisson(\lambda_i)$
  - $\lambda_i = \alpha P_i^{\beta}$
  - P is log population

Distances in thousands km



	Ml	Ti	SC	Ya	Fi	Tr	Ch	Mn	To	Ha
Malekula	0.0	0.5	0.6	4.4	1.2	2.0	3.2	2.8	1.9	5.7
Tikopia	0.5	0.0	0.3	4.2	1.2	2.0	2.9	2.7	2.0	5.3
Santa Cruz	0.6	0.3	0.0	3.9	1.6	1.7	2.6	2.4	2.3	5.4
Yap	4.4	4.2	3.9	0.0	5.4	2.5	1.6	1.6	6.1	7.2
Lau Fiji	1.2	1.2	1.6	5.4	0.0	3.2	4.0	3.9	0.8	4.9
Trobriand	2.0	2.0	1.7	2.5	3.2	0.0	1.8	0.8	3.9	6.7
Chuuk	3.2	2.9	2.6	1.6	4.0	1.8	0.0	1.2	4.8	5.8
Manus	2.8	2.7	2.4	1.6	3.9	0.8	1.2	0.0	4.6	6.7
Tonga	1.9	2.0	2.3	6.1	0.8	3.9	4.8	4.6	0.0	5.0
Hawaii	5.7	5.3	5.4	7.2	4.9	6.7	5.8	6.7	5.0	0.0

Neglects Spatial Autocorrelation / Neighboring Islands do trade



Images and example from: McElreath, R. (2018). Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC.

Islands – Taking the spatial correlation into account

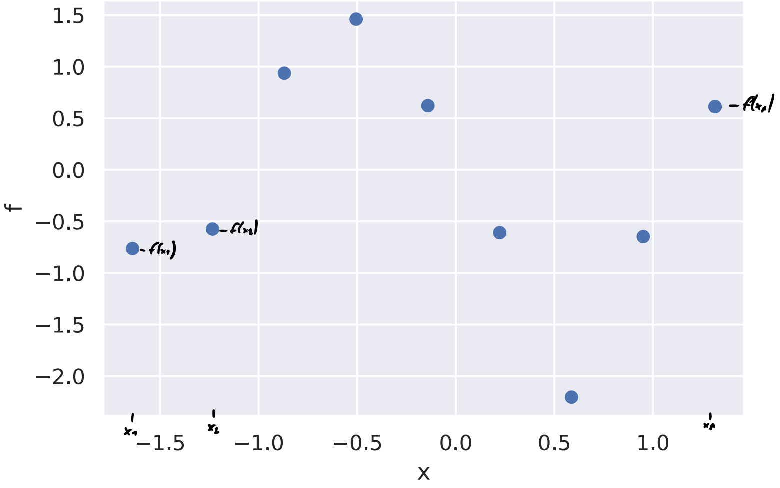
- First Model $i = 1,2,\dots,10$ for the 10 islands
  - $T_i \sim Poisson(\lambda_i)$
  - $\lambda_i = \alpha P_i^{\beta}$
- Taking
  - $\lambda_i = \exp(f_i)\,\alpha P_i^{\beta}$
- f_i works like a correction
  - $f_i = 0$ $\exp(0) = 1$ as expected
  - $f_i = -0.5$ $\exp(-0.5) = 0.6$ 60% of expected
  - $f_i = 0.25$ $\exp(-0.25) = 1.3$ 130% of expected
- Neighboring islands should have similar values of f
- How to model that ???

Definition of Gaussian Process (GP)

- Gaussian Process (GP) is a stochastic process (Collection of random variables)
- A GP is a distribution over functions of $f(x)$ if for any finite selection of points** $x_1, x_2, \dots, x_N$ the pdf $\mathrm{p}(f(x_1), f(x_2), \dots, f(x_N))$ is a multivariate Gaussian.*

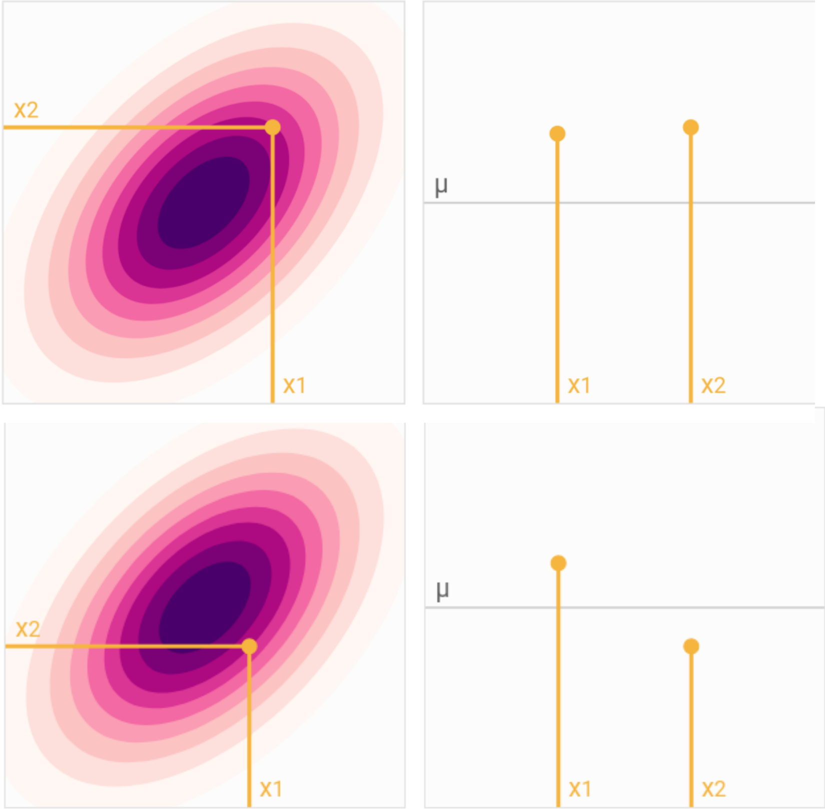
- MVGaussian are defined via mean and covariance matrix of f

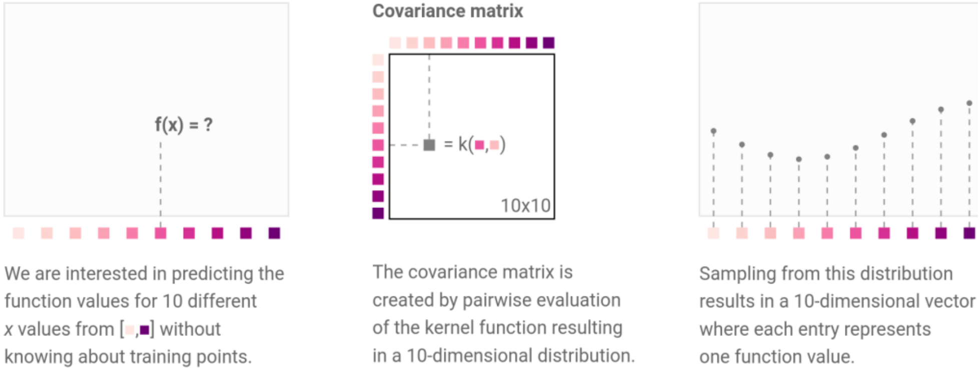
$$f \sim N(\mu, \Sigma)$$



*Definition of MacKay (notation adapted) **Of course $x_1$can be high dimensional data $x_i \in R^d$

Interpretation of f as a collection of random variables





We are interested in predicting the function values for 10 different *x* values from [■,■] without knowing about training points.

The covariance matrix is created by pairwise evaluation of the kernel function resulting in a 10-dimensional distribution.

Sampling from this distribution results in a 10-dimensional vector where each entry represents one function value.

Figures from: https://distill.pub/2019/visual-exploration-gaussian-processes/

Definition of Gaussian Process (GP)

- Use the GP to determine the parameters μ, Σ of the Multivariate Normal distribution $N(\mu, \Sigma)$
- GP thus defined by 2 functions
  - $m(x)$ the mean function produces the mean for every finite subset
  - $\mathrm{k}(x, x_*)$ the Kernel function produces the covariance matrix for every finite subset
- New datapoint ➔ increase dimension of Multivariate Normal distribution
  - ➔ We need a function to create parameters

$$f(x_*) \sim GP\big(m(x), k(x, x_*)\big)$$

$m(x)$ is usually zero (We can add such offsets)

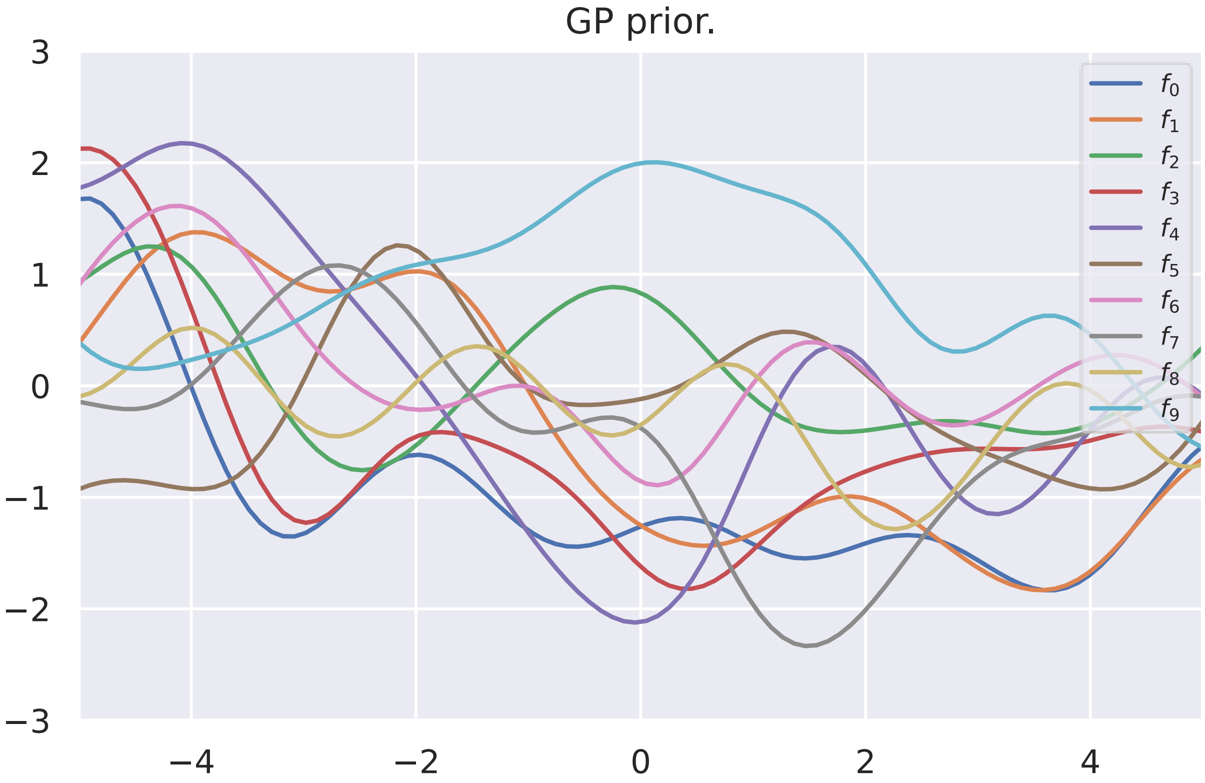
$k(x, x_*)$ Kernel

$$f \sim N(0, [K(X, X)])$$

$$K(X, X) = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_n, x_1) \\ \vdots & \ddots & \vdots \\ k(x_1, x_n) & \cdots & k(x_n, x_n) \end{bmatrix}$$

Samples from prior

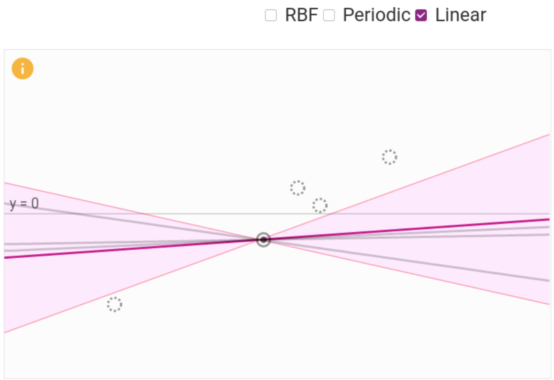
- Prior $p(f\,|X)$
- With rbf-kernel

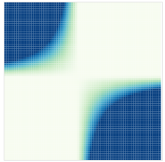


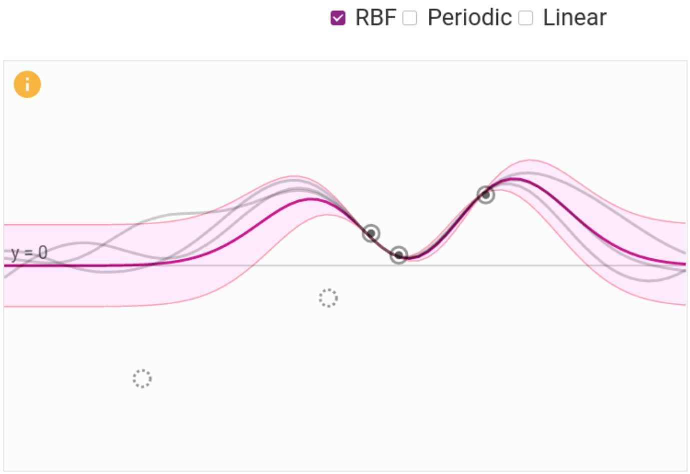
Kernel/ covariance function

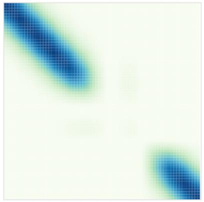
- Linear kernel $k(x, x_*) = \langle x, x_* \rangle$
- Polynomial kernel $k(x, x_*) = \langle x, x_* \rangle^d$
- Radial basis function kernel

$$k(x, x_*) = \sigma_f^2 \cdot e^{\left(-\frac{\|x - x_*\|^2}{2l^2}\right)}$$


□ RBF □ Periodic ☑ Linear




☑ RBF □ Periodic □ Linear

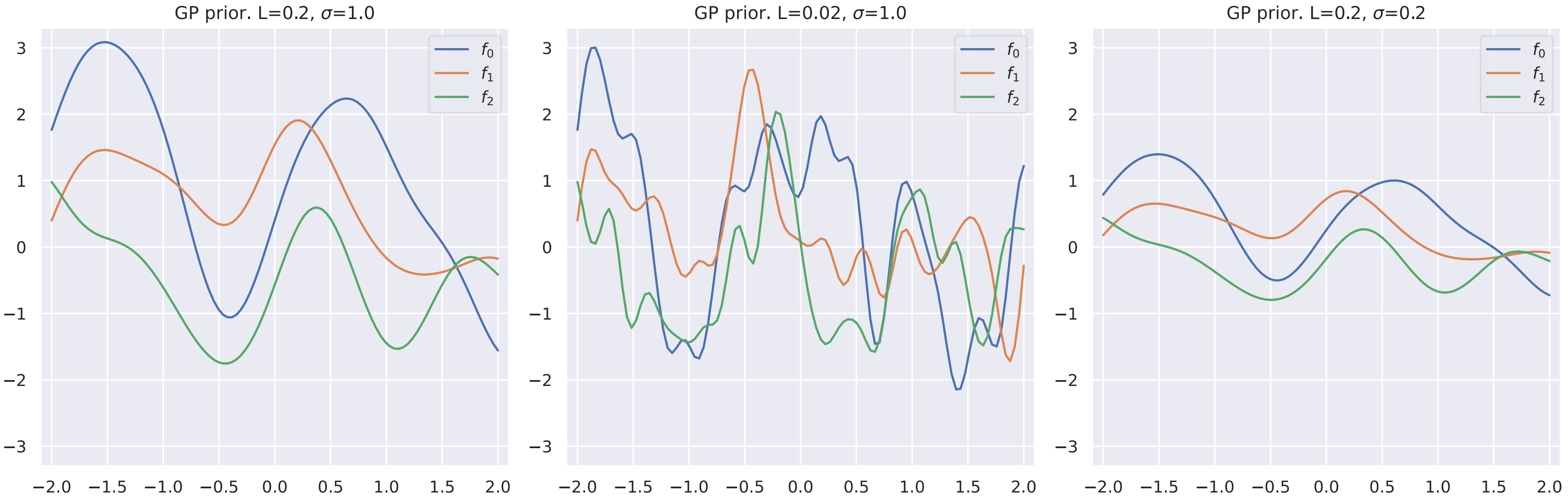


Stationary	Non-Stationary
~~global trend~~	global trend

Figures from: https://distill.pub/2019/visual-exploration-gaussian-processes/

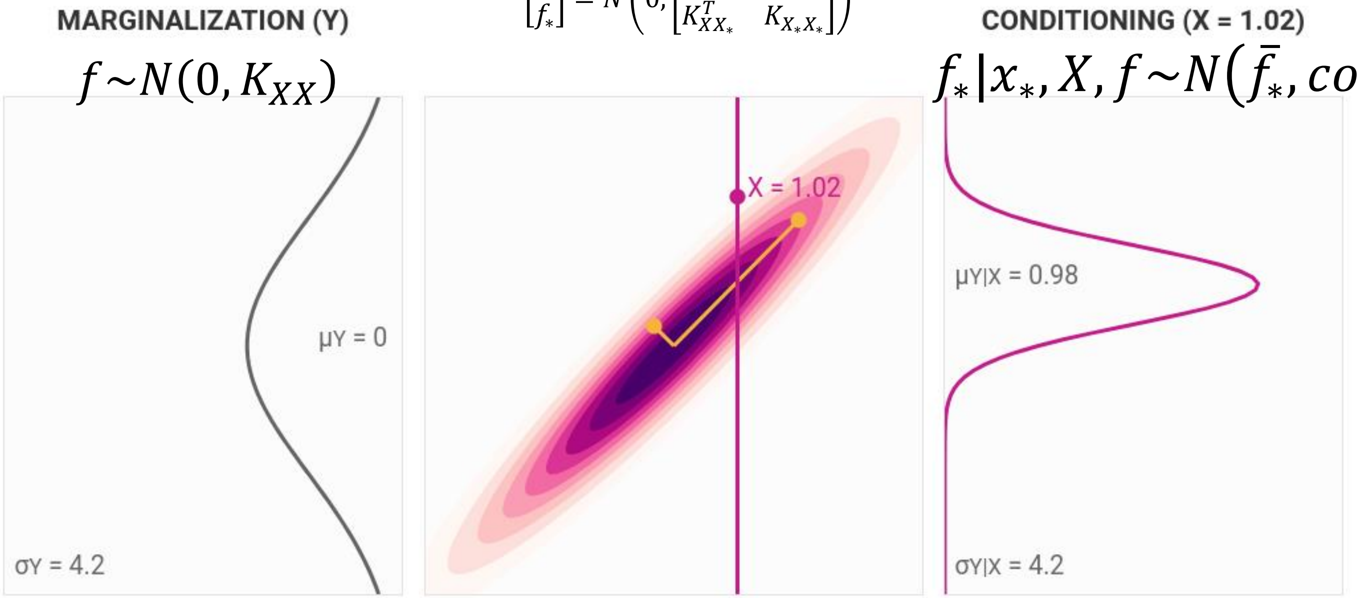
Effect of kernel parameters

$$k(x, x_*) = \sigma^2 \exp\left(-\frac{1}{2}(x - x_*)^T L (x - x_*)\right)$$



Conditioning and Marginalization

$$\begin{bmatrix} f \\ f_* \end{bmatrix} = N\left(0, \begin{bmatrix} K_{XX} & K_{XX_*} \\ K_{XX_*}^T & K_{X_*X_*} \end{bmatrix}\right)$$



$$f \sim N(0, K_{XX})$$

$$f_* | x_*, X, f \sim N\left(\bar{f}_*, cov(f_*)\right)$$

A bivariate normal distribution in the center. On the left you can see the result of marginalizing this distribution for Y, akin to integrating along the X axis. On the right you can see the distribution conditioned on a given X, which is similar to a cut through the original distribution. The Gaussian distribution and the conditioned variable can be changed by dragging the handles.

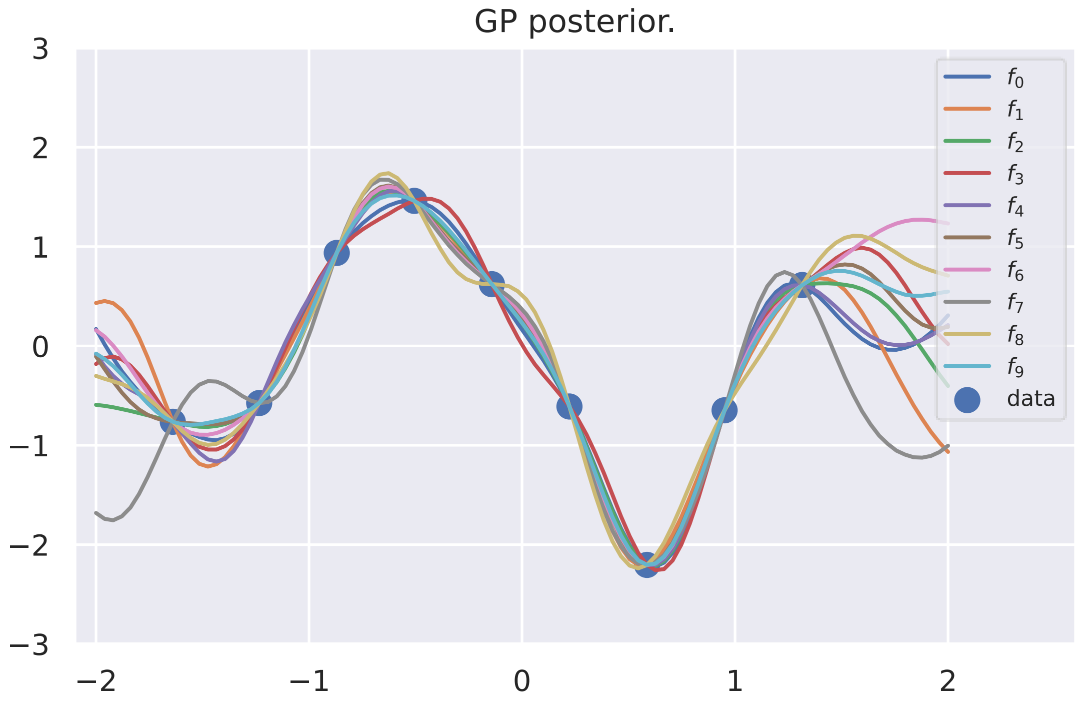
Figure from: https://distill.pub/2019/visual-exploration-gaussian-processes/

Posterior samples/ Conditioning

$$p(f, f_*) \sim N\left(0, \begin{bmatrix} K_{XX} & K_{XX_*} \\ K_{XX_*}^T & K_{X_*X_*} \end{bmatrix}\right)$$

$$p(f_*|f) = \frac{p(f_*, f)}{p(f)}$$

$$\underbrace{\boldsymbol{p(f_*|x_*, X, f)}}_{\text{conditioning}} \sim N\left(\bar{f}_*, cov(f_*)\right)$$



Algorithms

- MCMC (Neal 1997; Christensen et al. 2006)
- variational (Girolami and Rogers 2006; Opper and Archambeau 2009)
- expectation propagation (Kuss and Rasmussen 2005; Nickisch and Rasmussen 2008)
- Gaussian approximation (Rasmussen 2006)
- Analytic

Use MCMC to solve GP

Full Bayes

Define prior:

- $\rho \sim \mathrm{InvGamma}(3{,}1)$
- $\sigma_f \sim N(0{,}1)$

Define GP

- $f \sim \text{multivariate normal}\left(0, K\left(x \middle| \sigma_f, \rho\right)\right)$

```
data {
  int<lower=1> N;
  real x[N];
  vector[N] f;
}
transformed data {
  vector[N] mu = rep_vector(0, N);
}
parameters {
  real<lower=0> rho;
  real<lower=0> sigma_f;
}
model {
  matrix[N, N] L_K;
  matrix[N, N] K = cov_exp_quad(x, sigma_f, rho);
  L_K = cholesky_decompose(K);  ↖ O(N^3)

  rho ~ inv_gamma(3, 1);
  sigma_f ~ std_normal();
  f ~ multi_normal_cholesky(mu, L_K);
}
```

Use MCMC to solve GP with (noisy) observations

Define prior:

$\rho \sim \mathrm{InvGamma}(3,1)$
$\sigma_f \sim N(0,1)$
$\sigma_n \sim N(0,0.1)$

Define GP

$f \sim \text{multivariate normal}\left(0, K\left(x \middle| \sigma_f, \rho\right)\right)$

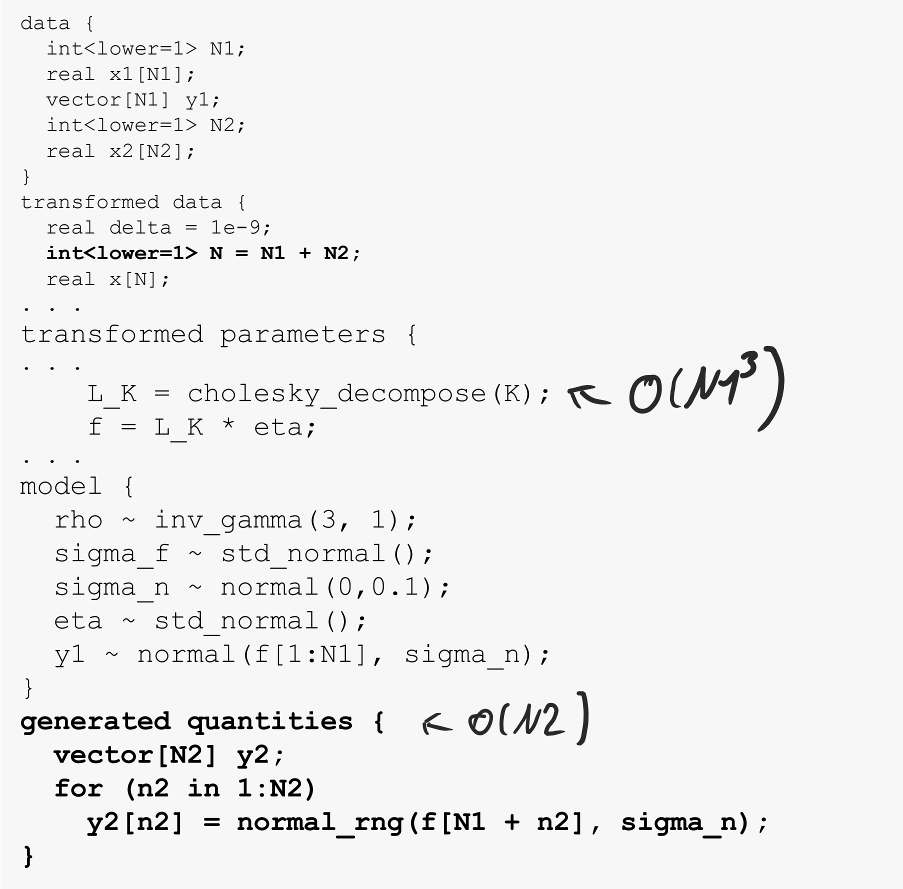
Define likelihood

$y_i \sim N(f_i, \sigma_n) \forall i \in \{1, \ldots, N\}$*

```
data {
  int<lower=1> N;
  real x[N];
  vector[N] y;
}
transformed data {
  real delta = 1e-9;
}
parameters {
  real<lower=0> rho;
  real<lower=0> sigma_f;
  real<lower=0> sigma_n;
  vector[N] eta;
}
model {
  vector[N] f;
  {
    matrix[N, N] L_K;
    matrix[N, N] K = cov_exp_quad(x, sigma_f, rho);
    for (n in 1:N) // diagonal elements
      K[n, n] = K[n, n] + delta;
    L_K = cholesky_decompose(K);
    f = L_K * eta;
  }
  rho ~ inv_gamma(3, 1);
  sigma_f ~ std_normal();
  sigma_n ~ normal(0,0.1);
  eta ~ std_normal();
  y ~ normal(f, sigma_n);
}
```

*We can also marginalize out f to include the noise in the multivariate Gaussian directly (replace $K(X,X) = K(X,X) + \sigma_n^2 I$)

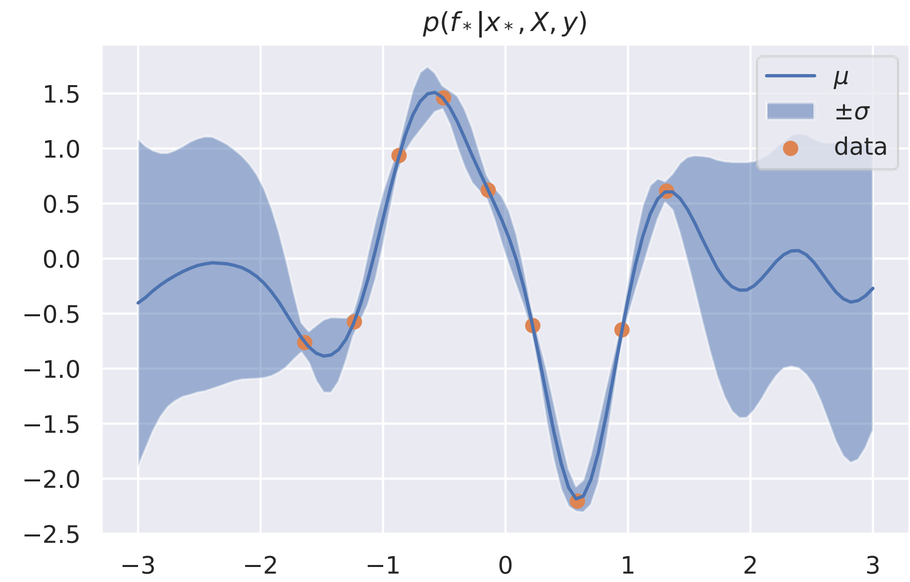
Inference with MCMC $p(f_*|x_*, X, f)$



```
data {
  int<lower=1> N1;
  real x1[N1];
  vector[N1] y1;
  int<lower=1> N2;
  real x2[N2];
}
transformed data {
  real delta = 1e-9;
  int<lower=1> N = N1 + N2;
  real x[N];
. . .
transformed parameters {
. . .
     L_K = cholesky_decompose(K);
     f = L_K * eta;
. . .
model {
  rho ~ inv_gamma(3, 1);
  sigma_f ~ std_normal();
  sigma_n ~ normal(0,0.1);
  eta ~ std_normal();
  y1 ~ normal(f[1:N1], sigma_n);
}
generated quantities {
  vector[N2] y2;
  for (n2 in 1:N2)
    y2[n2] = normal_rng(f[N1 + n2], sigma_n);
}
```

$\leftarrow O(N1^3)$

$\leftarrow O(N2)$



Complexity:

- Create Samples: $O(\#S \cdot N1^3)$

- Prediction: $O(\#S \cdot N2)$*

*new point -> need to create new samples $\max\left(O(\#S \cdot N2), O(\#S \cdot \mathrm{N}1^3)\right)$

Islands – Advantage of MCMC

Can model:

- Prior-distributions on hyper-parameters
- Any kind of likelihood function

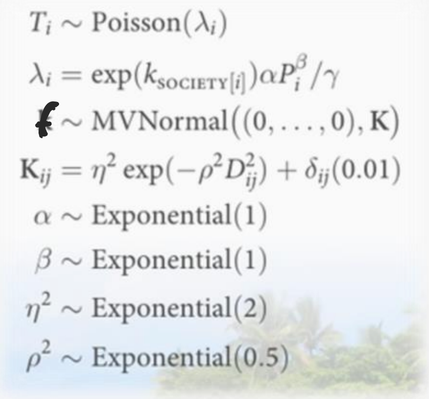
```
data {
  int<lower=1> N;
  matrix D[N,N];
  vector[N] T;
  vector[N] P;}
transformed data {
  real delta = 1e-9;}
parameters {
  real<lower=0> rho;
  real<lower=0> sigma_f;
  real<lower=0> alpha;
  real<lower=0> beta;
  vector[N] eta;}
model {
  vector[N] lambda;{
    matrix[N, N] L_K;
    for (i in 1:(N - 1)) {
      K[i, i] = 1 + delta;
        for (j in (i + 1):N) {
          K[i, j] = sigma_f * exp(-square(rho*D[i,j]));
          K[j, i] = K[i, j];}}
    K[N, N] = 1 + delta;
    f = cholesky_decompose(K) * eta;
    lambda = exp(f)*alpha*pow(P,beta)
  }
  alpha ~ exponential(1);
  beta ~ exponential(1);
  rho ~ exponential(0.5);
  sigma_f ~ exponential(2);
  T ~ poisson(lambda);
}
```

Kernel →

inv. Link →

likelihood →

} priors



$$T_i \sim \text{Poisson}(\lambda_i)$$
$$\lambda_i = \exp(k_{\text{SOCIETY}[i]})\alpha P_i^{\beta}/\gamma$$
$$f \sim \text{MVNormal}((0,\dots,0),\mathbf{K})$$
$$\mathbf{K}_{ij} = \eta^2\exp(-\rho^2 D_{ij}^2) + \delta_{ij}(0.01)$$
$$\alpha \sim \text{Exponential}(1)$$
$$\beta \sim \text{Exponential}(1)$$
$$\eta^2 \sim \text{Exponential}(2)$$
$$\rho^2 \sim \text{Exponential}(0.5)$$

Images and example from: McElreath, R. (2018). Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC.

GP with non-Gaussian likelihood

No prior-distribution for hyper-parameters

1. Compute a posterior predictive dist (with the approximated posterior)

$$p(f_*|x_*,X,y) = \int p(f_*|x_*,X,y,f) \cdot p(f|X,y)\,df \quad \leftarrow \text{usually not Gaussian}$$

$$\text{if all Gaussian} = N\left(K_{X_*X}K_{XX}^{-1}\overset{\leftarrow O(N^3)}{f}, K_{X_*X_*} - K_{X_*X}K_{XX}^{-1}K_{XX_*}\right)$$

2. (Marginalize out f_* to produce a probabilistic prediction)

$$\mathrm{p}(y_*|x_*,X,y) = \int \mathrm{inv\_link}(f_*)p(f_*|x_*,X,y)\,df_*$$

Variational Gaussian Process

- Compared to MCMC we “move” the data into a variational distribution $q(f) \sim N(\mu, \Sigma)$
  - We don’t have to create new samples for new predictions

$$p(f_*|x_*, X, y) = \int p(f_*|x_*, X, f) \cdot \overset{\cancel{p(f|X,y)}}{q(f)}\, df$$

- Algo:
  - Replace $p(f|X, y) \approx q(f) \sim \mathrm{N}(\mu, \Sigma)$
    - Variational parameters e.g., $\{\mu, \Sigma, \rho, \sigma_f\}$
  - Minimize $\mathrm{KL}[q(f)||p(f|X, y)]$
    - Maximize *ELBO** $\underbrace{\int \log\big(p(y|f)\big) q(f)\, df}_{N \text{ single-dimensional integration}} - \underbrace{\mathrm{KL}[q(f)||p(f)]}_{\text{analytic}}$

$$\mu, \mathrm{diag}(\Sigma) \rightleftarrows \mu, \Sigma, \rho, \sigma_f$$

*Evidence Lower Bound (ELBO)

Spares variational Gaussian Process (SVGP)

- Reduce K_{XX}^{-1} inversion complexity $O(N)$ to $O(N_u)$
- Add inducing points u and define var dist $q(f_u) \sim N(\mu, \Sigma)$

$$p(f|f_u)q(f_u) = q(f, f_u) \sim N\left(0, \begin{bmatrix} K_{XX} & K_{XX_u} \\ K_{X_uX} & K_{X_uX_u} \end{bmatrix}\right)$$

- Maximize *ELBO*

$$\int \log(p(y|f)) \underbrace{\int q(f, f_u) df_u}_{p(f)} df - \mathrm{KL}[q(f_u)||p(f_u)]$$

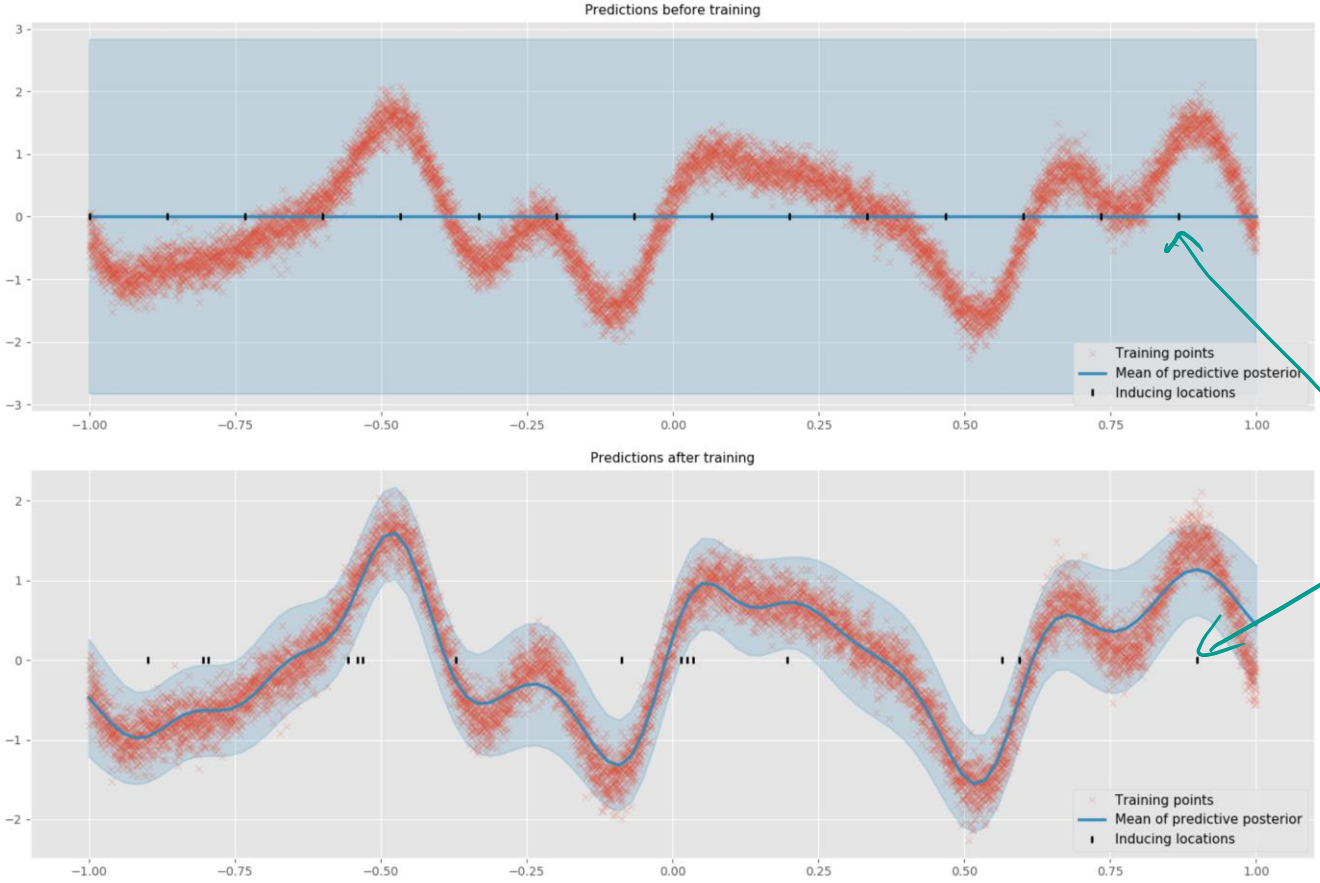
- Prediction

$$p(f_*|x_*, X, y) = \int p(f_*|x_*, X, f_u) \cdot q(f_u) df_u$$

$$\begin{bmatrix} f \\ f_u \\ f_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K_{XX} & K_{XX_u} & K_{XX_*} \\ K_{XX_u}^T & K_{X_uX_u} & K_{X_uX_*} \\ K_{XX_*}^T & K_{X_uX_*}^T & K_{X_*X_*} \end{bmatrix}\right)$$

SVGP was first introduced by: Titsias, M. (2009). Variational Learning of Inducing Variables in Sparse Gaussian Processes. Artificial Intelligence and Statistics
Extended by: Hensman et al., 2013/ 2015

Spares variational Gaussian Process (SVGP)



Inducing points

Figure from: Sparse and Variational Gaussian Process (SVGP)—What To Do When Data is Large | by Wei Yi | Towards Data Science. (o. J.). Abgerufen 9. Juni 2021, von https://towardsdatascience.com/sparse-and-variational-gaussian-process-what-to-do-when-data-is-large-2d3959f430e7

Spares variational Gaussian Process (SVGP)

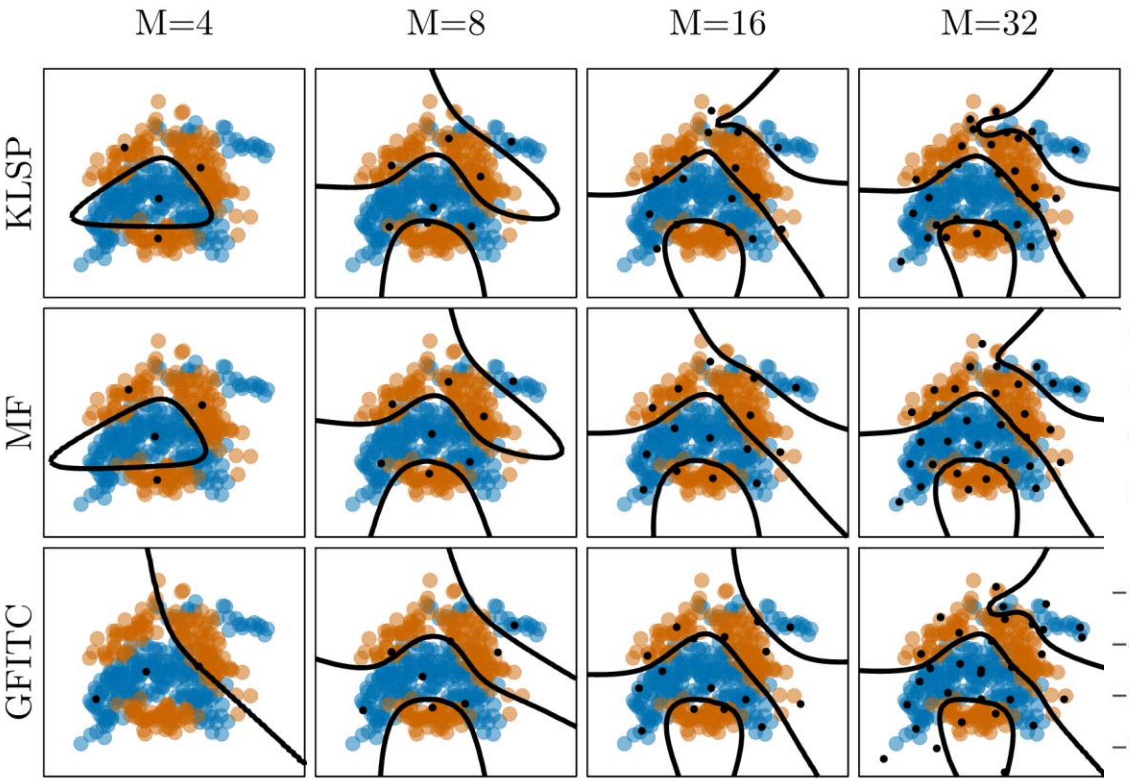


Figure from *[1]

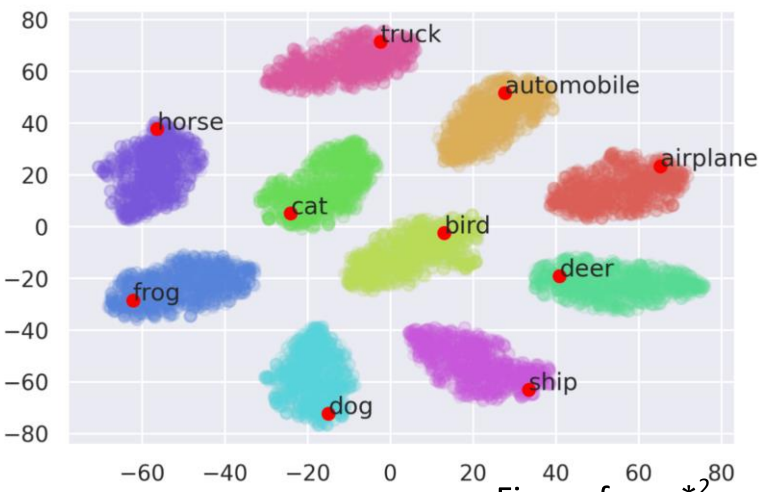


Figure from *[2]

*[1]: Hensman, J., Matthews, A., & Ghahramani, Z. (2015). Scalable Variational Gaussian Process Classification. Artificial Intelligence and Statistics, 351–360. http://proceedings.mlr.press/v38/hensman15.html

*[2]: van Amersfoort, J., Smith, L., Jesson, A., Key, O., & Gal, Y. (2021). Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression. http://arxiv.org/abs/2102.11409

GPC (Gaussian / Laplace approximation)

No prior-distribution for hyper-parameters

$$p(f_*|x_*,X,y) = \int p(f_*|x_*,X,f)\cdot q(f|X,y)df\,, \qquad q(f|X,y)\sim N\big(f|\hat{f},A^{-1}\big)w$$

$$q(\mathbf{f}|X,\mathbf{y}) \;=\; \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}},A^{-1}) \;\propto\; \exp\big(-\tfrac{1}{2}(\mathbf{f}-\hat{\mathbf{f}})^\top A(\mathbf{f}-\hat{\mathbf{f}})\big),$$

where $\hat{\mathbf{f}} = \operatorname{argmax}_{\mathbf{f}}\, p(\mathbf{f}|X,\mathbf{y})$ and $A = -\nabla\nabla \log p(\mathbf{f}|X,\mathbf{y})|_{\mathbf{f}=\hat{\mathbf{f}}}$ is the Hessian of

Laplace approx (Second order Taylor)

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

Gaussian Process analytic

No prior distributions

All distributions are Gaussian

$$\begin{bmatrix} y \\ f_* \end{bmatrix} = N\left(0, \begin{bmatrix} K(X,X) + \sigma_n I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

Conditioning

$$p(f_*|x_*,X,y) \sim N\left(\bar{f}_*, cov(f_*)\right), \qquad y \sim N(f(x), \sigma_n)$$

$$\bar{f}_* \triangleq \mathbb{E}[f_*|x_*,X,y] = K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} y$$

$$cov(f_*) = K(X_*,X_*) - K(K_*,X)[K(X,X) + \sigma_n^2 I]^{-1} K(X,X_*)$$

Kernel parameters estimation

$$k(\mathbf{x}_p, \mathbf{x}_q) \;=\; \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^\top M (\mathbf{x}_p - \mathbf{x}_q)\right) + \sigma_n^2 \delta_{pq},$$

Optimize θ with **marginal likelihood**, $\theta = \{\rho, \sigma_f^2, \sigma_n^2, M\}$

$$p(f|X,y,\theta) = \frac{p(y|f,X,\theta)p(f|X,\theta)}{\underbrace{\int p(y|f,X,\theta)p(f|X,\theta)df}}$$

$$\log p(\mathbf{y}|X,\boldsymbol{\theta}) \;=\; -\frac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi$$

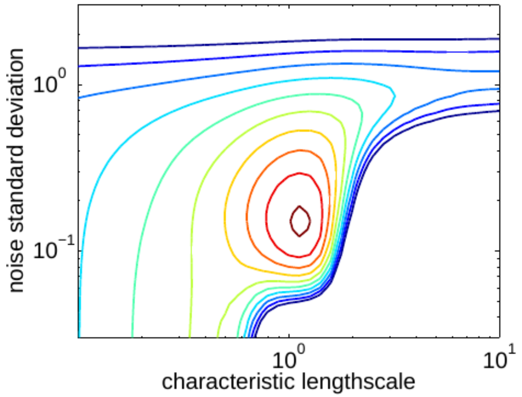
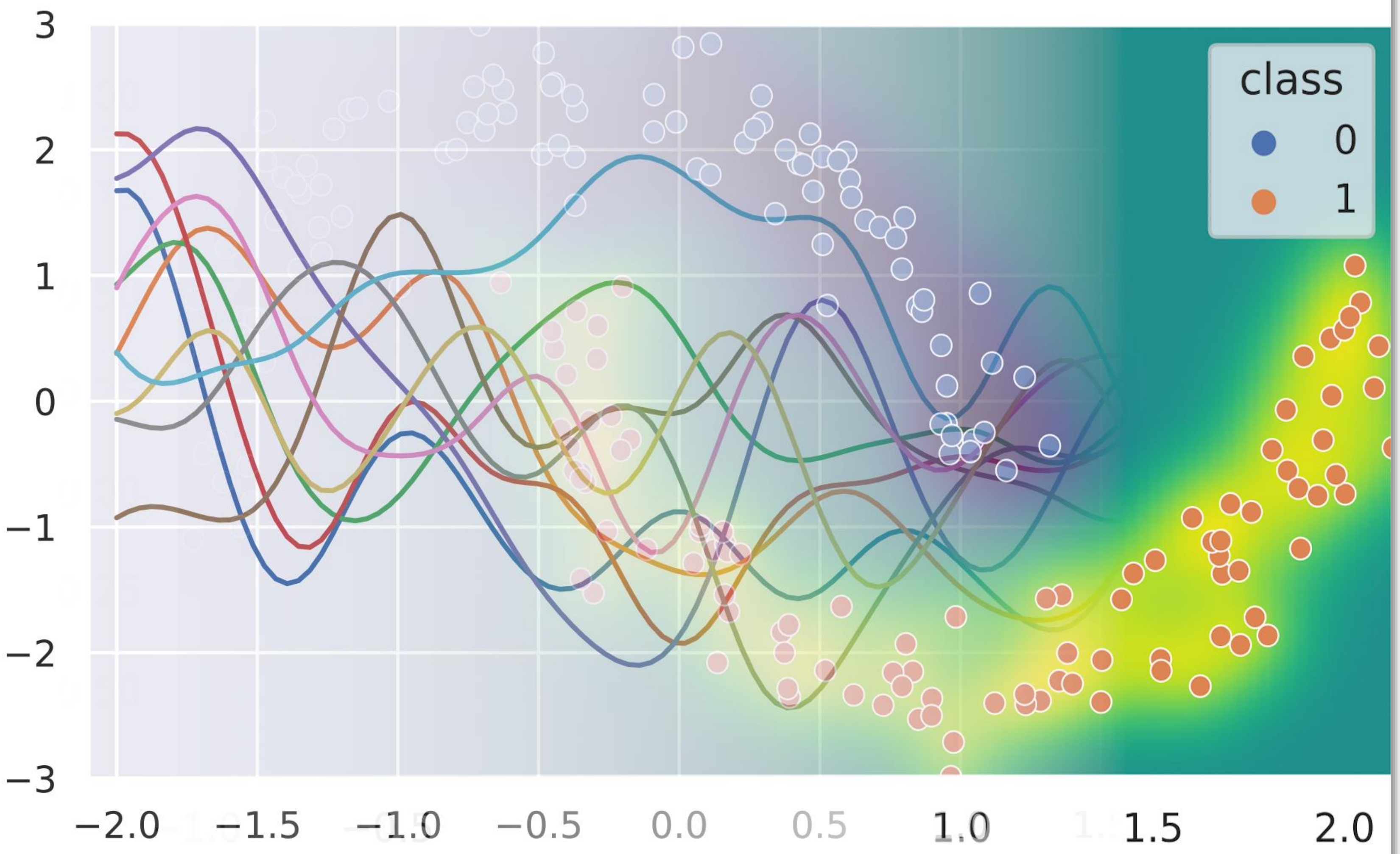


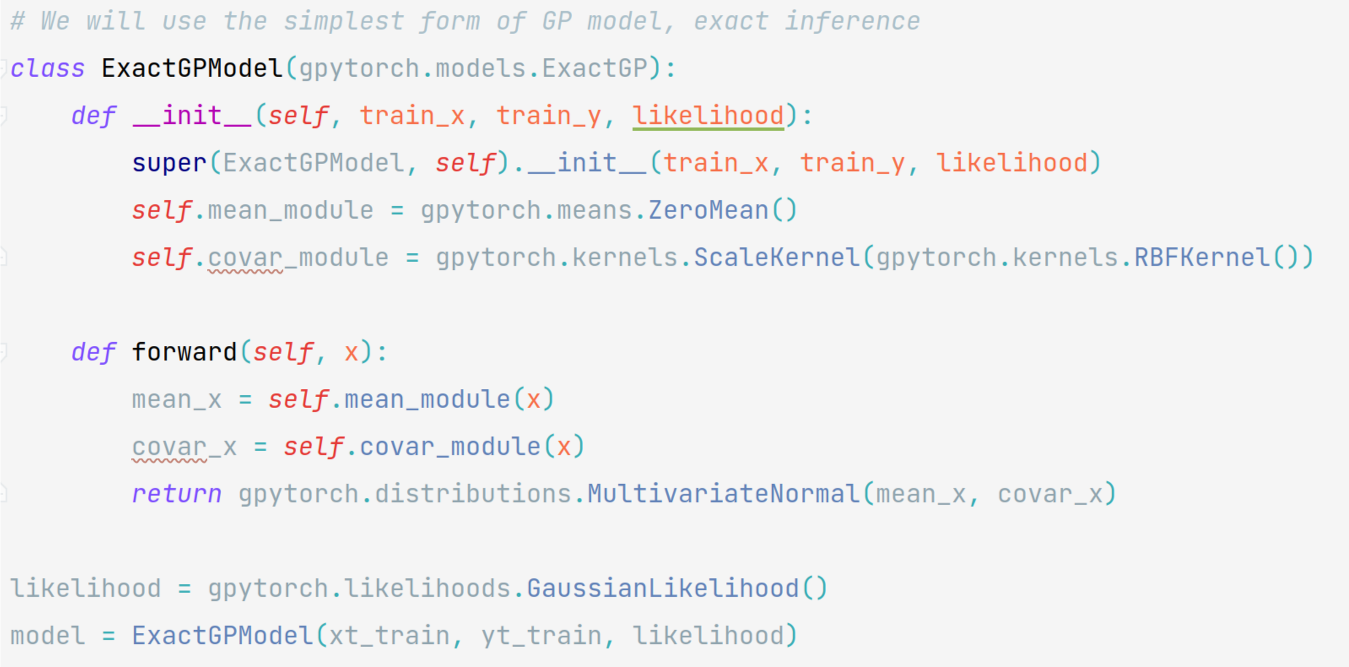
Figure from: C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006


3
2
1
0
−1
−2
−3
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
class
0
1

Thanks for your attention

GPR with GPytorch

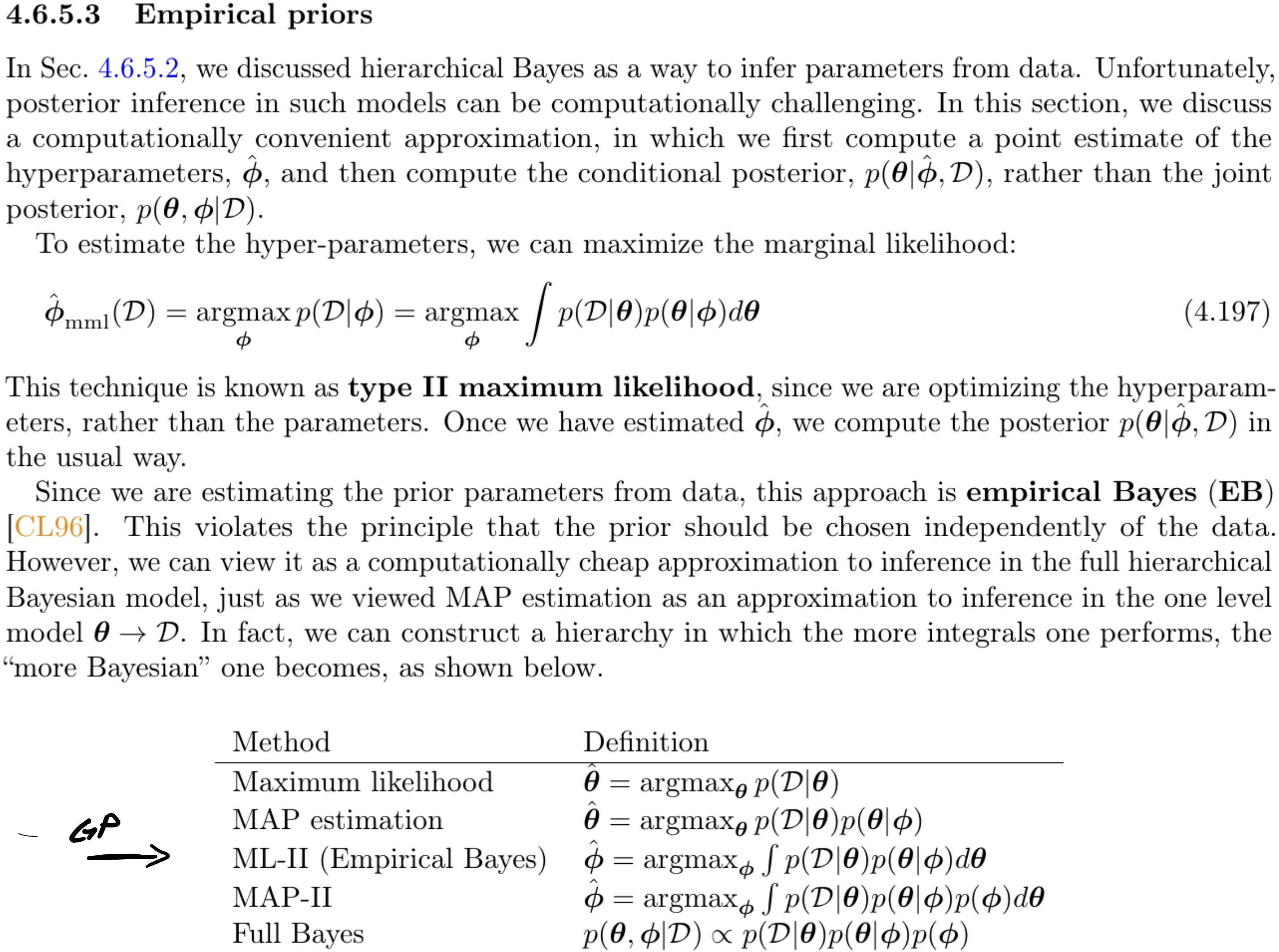
1. GP Model
2. Likelihood
3. Prior mean
4. Kernel/ Prior covariance
5. MultivariateNormal Distribution



```python
# We will use the simplest form of GP model, exact inference
class ExactGPModel(gpytorch.models.ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super(ExactGPModel, self).__init__(train_x, train_y, likelihood)
        self.mean_module = gpytorch.means.ZeroMean()
        self.covar_module = gpytorch.kernels.ScaleKernel(gpytorch.kernels.RBFKernel())

    def forward(self, x):
        mean_x = self.mean_module(x)
        covar_x = self.covar_module(x)
        return gpytorch.distributions.MultivariateNormal(mean_x, covar_x)

likelihood = gpytorch.likelihoods.GaussianLikelihood()
model = ExactGPModel(xt_train, yt_train, likelihood)
```



4.6.5.3 Empirical priors

In Sec. 4.6.5.2, we discussed hierarchical Bayes as a way to infer parameters from data. Unfortunately, posterior inference in such models can be computationally challenging. In this section, we discuss a computationally convenient approximation, in which we first compute a point estimate of the hyperparameters, $\hat{\boldsymbol{\phi}}$, and then compute the conditional posterior, $p(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}, \mathcal{D})$, rather than the joint posterior, $p(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathcal{D})$.

To estimate the hyper-parameters, we can maximize the marginal likelihood:

$$\hat{\boldsymbol{\phi}}_{\text{mml}}(\mathcal{D}) = \underset{\boldsymbol{\phi}}{\operatorname{argmax}}\, p(\mathcal{D}|\boldsymbol{\phi}) = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\phi})d\boldsymbol{\theta} \tag{4.197}$$

This technique is known as **type II maximum likelihood**, since we are optimizing the hyperparameters, rather than the parameters. Once we have estimated $\hat{\boldsymbol{\phi}}$, we compute the posterior $p(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}, \mathcal{D})$ in the usual way.

Since we are estimating the prior parameters from data, this approach is **empirical Bayes** (**EB**) [CL96]. This violates the principle that the prior should be chosen independently of the data. However, we can view it as a computationally cheap approximation to inference in the full hierarchical Bayesian model, just as we viewed MAP estimation as an approximation to inference in the one level model $\boldsymbol{\theta} \to \mathcal{D}$. In fact, we can construct a hierarchy in which the more integrals one performs, the "more Bayesian" one becomes, as shown below.

Method	Definition
Maximum likelihood	$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{D}\|\boldsymbol{\theta})$
MAP estimation	$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\phi})$
ML-II (Empirical Bayes)	$\hat{\boldsymbol{\phi}} = \operatorname{argmax}_{\boldsymbol{\phi}} \int p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\phi})d\boldsymbol{\theta}$
MAP-II	$\hat{\boldsymbol{\phi}} = \operatorname{argmax}_{\boldsymbol{\phi}} \int p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\phi})p(\boldsymbol{\phi})d\boldsymbol{\theta}$
Full Bayes	$p(\boldsymbol{\theta}, \boldsymbol{\phi}\|\mathcal{D}) \propto p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\phi})p(\boldsymbol{\phi})$

GP →