Introduction to Nonparametric Bayesian Modeling and Gaussian Process Regression

Piyush Rai

Dept. of CSE, IIT Kanpur

(Mini-course: lecture 3)

Nov 07, 2015

Recap

Optimization vs Inference

All ML problems require estimating parameters given data. Primarily two views:

1. Learning as Optimization

- Parameter θ is a fixed unknown
- Seeks a point estimate (single best answer) for θ

$$\hat{\theta} = \arg\min_{\theta} \text{Loss}(\mathcal{D}; \theta) \qquad \text{subject to constraints on } \theta$$

- Probabilistic methods such as MLE and MAP also fall in this category

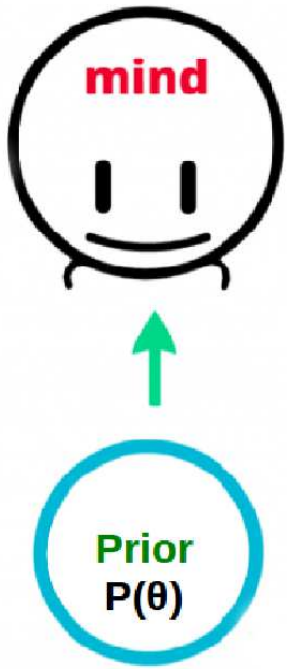
2. Learning as (Bayesian) Inference

- Parameter θ is a random variable with a prior distribution $P(\theta)$
- Seeks a posterior distribution over the parameters

$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta) P(\theta)}{P(\mathcal{D})}$$

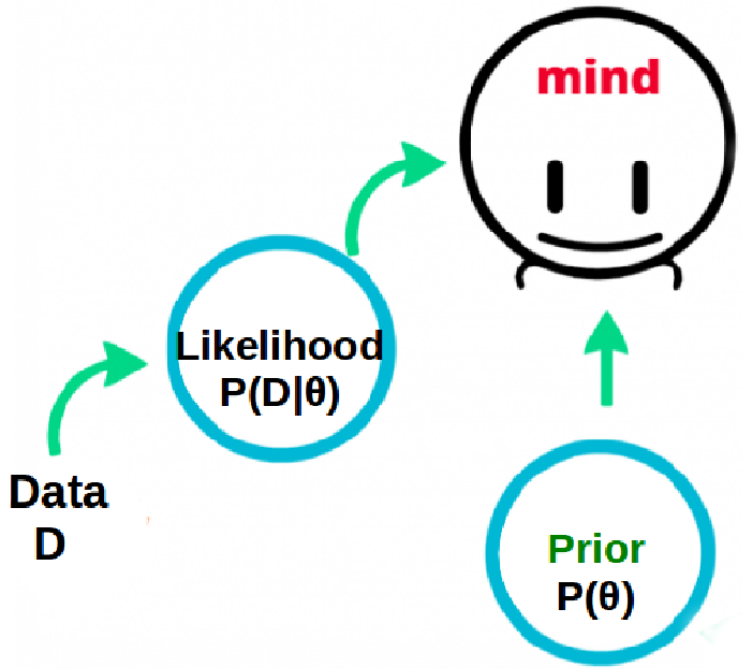
Bayesian Learning

- Prior distribution specifies our prior belief/knowledge about parameters θ
- Bayesian inference updates the prior and gives the posterior



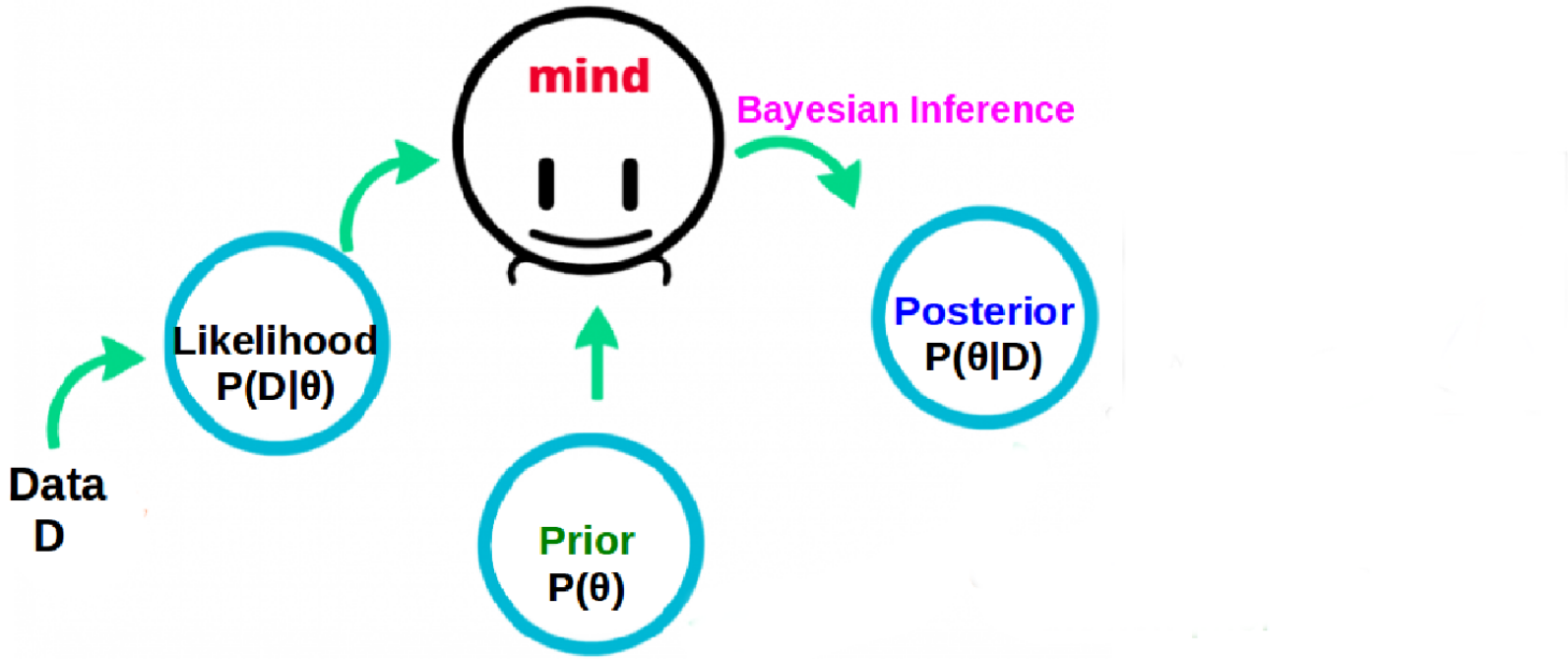
Bayesian Learning

- Prior distribution specifies our prior belief/knowledge about parameters θ
- Bayesian inference updates the prior and gives the posterior



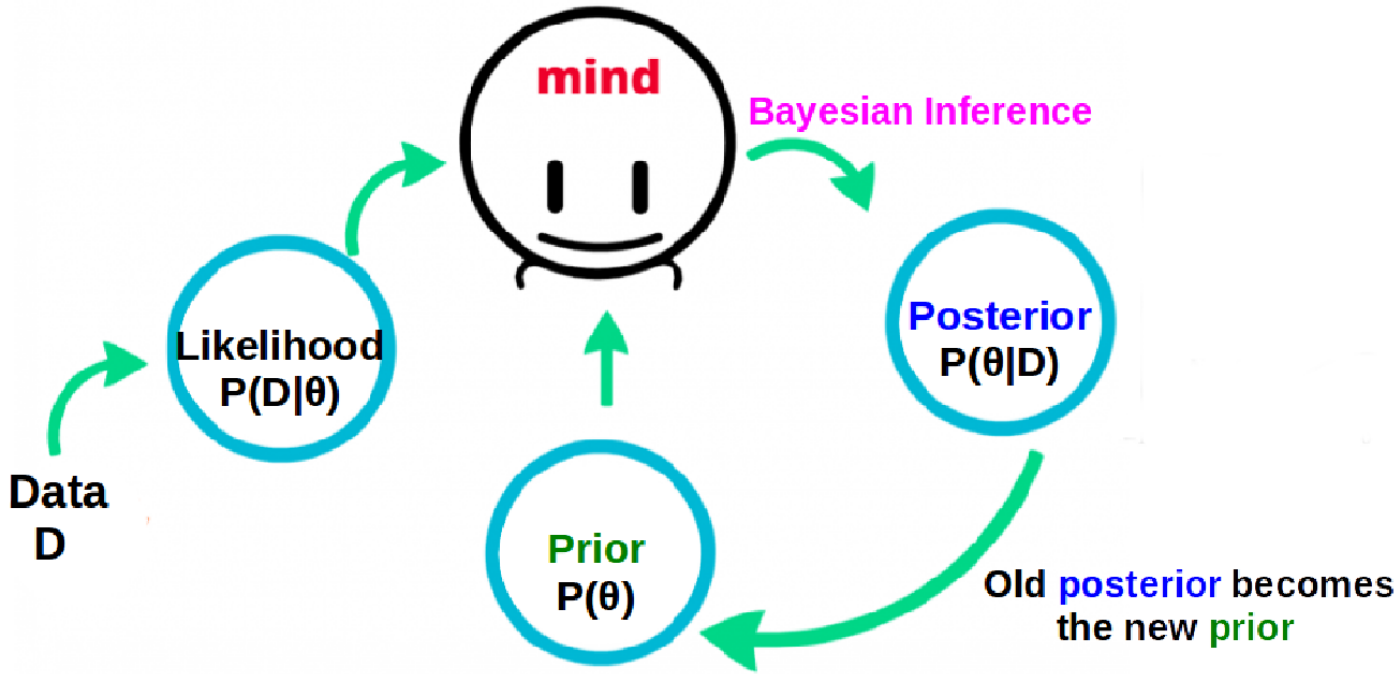
Bayesian Learning

- Prior distribution specifies our prior belief/knowledge about parameters θ
- Bayesian inference updates the prior and gives the posterior



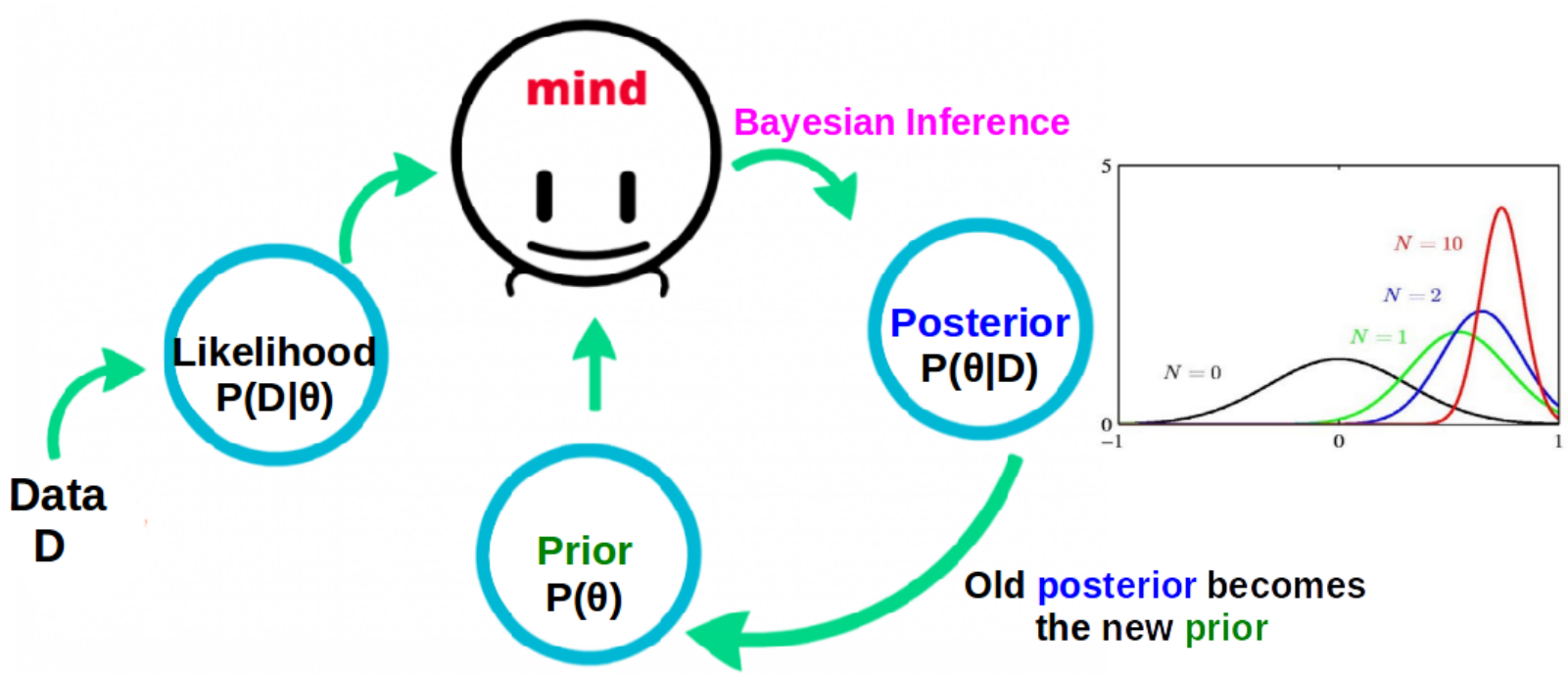
Bayesian Learning

- Prior distribution specifies our prior belief/knowledge about parameters θ
- Bayesian inference updates the prior and gives the posterior



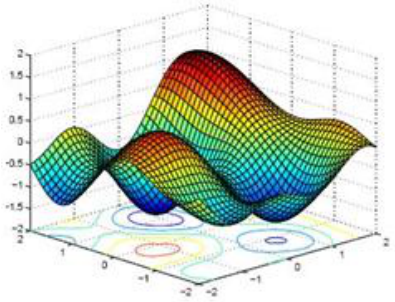
Bayesian Learning

- Prior distribution specifies our prior belief/knowledge about parameters θ
- Bayesian inference updates the prior and gives the posterior



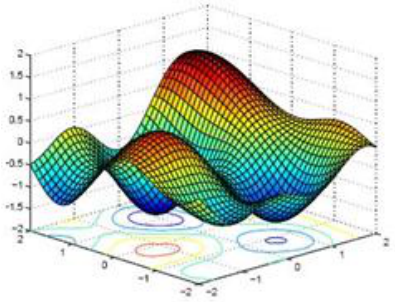
Why be Bayesian?

- Posterior $P(\theta|\mathcal{D})$ **quantifies uncertainty** in the parameters



Why be Bayesian?

- Posterior $P(\theta|\mathcal{D})$ **quantifies uncertainty** in the parameters

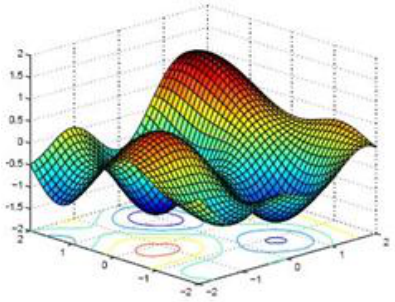


- More **robust predictions** by averaging over the posterior $P(\theta|\mathcal{D})$

$$P(d_{test}|\hat{\theta}) \quad \text{vs} \quad P(d_{test}|\mathcal{D}) = \int P(d_{test}|\theta)P(\theta|\mathcal{D})d\theta$$

Why be Bayesian?

- Posterior $P(\theta|\mathcal{D})$ **quantifies uncertainty** in the parameters



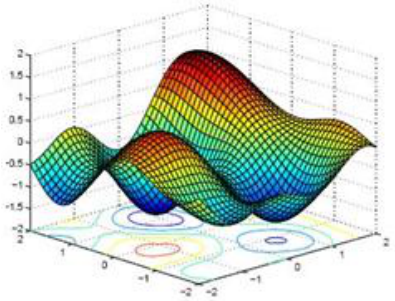
- More **robust predictions** by averaging over the posterior $P(\theta|\mathcal{D})$

$$P(d_{test}|\hat{\theta}) \quad \text{vs} \quad P(d_{test}|\mathcal{D}) = \int P(d_{test}|\theta)P(\theta|\mathcal{D})d\theta$$

- Allows inferring hyperparameters of the model and doing model comparison

Why be Bayesian?

- Posterior $P(\theta|\mathcal{D})$ **quantifies uncertainty** in the parameters



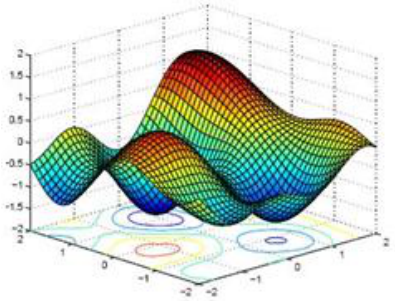
- More **robust predictions** by averaging over the posterior $P(\theta|\mathcal{D})$

$$P(d_{test}|\hat{\theta}) \quad \text{vs} \quad P(d_{test}|\mathcal{D}) = \int P(d_{test}|\theta)P(\theta|\mathcal{D})d\theta$$

- Allows inferring hyperparameters of the model and doing model comparison
- Offers a natural way for informed data acquisition (active learning)
  - Can use the predictive posterior of unseen data points to guide data selection

Why be Bayesian?

- Posterior $P(\theta|\mathcal{D})$ **quantifies uncertainty** in the parameters



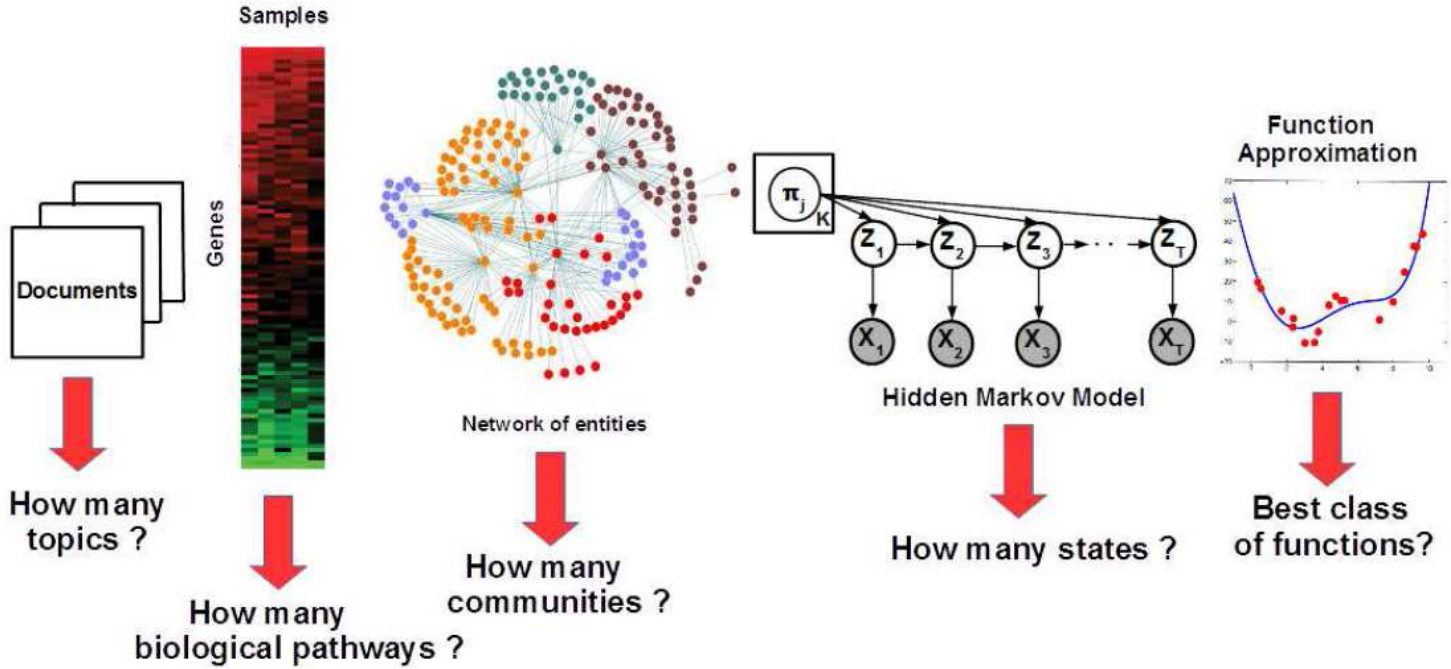
- More **robust predictions** by averaging over the posterior $P(\theta|\mathcal{D})$

$$P(d_{test}|\hat{\theta}) \quad \text{vs} \quad P(d_{test}|\mathcal{D}) = \int P(d_{test}|\theta)P(\theta|\mathcal{D})d\theta$$

- Allows inferring hyperparameters of the model and doing model comparison
- Offers a natural way for informed data acquisition (active learning)
  - Can use the predictive posterior of unseen data points to guide data selection
- Can do nonparametric Bayesian modeling

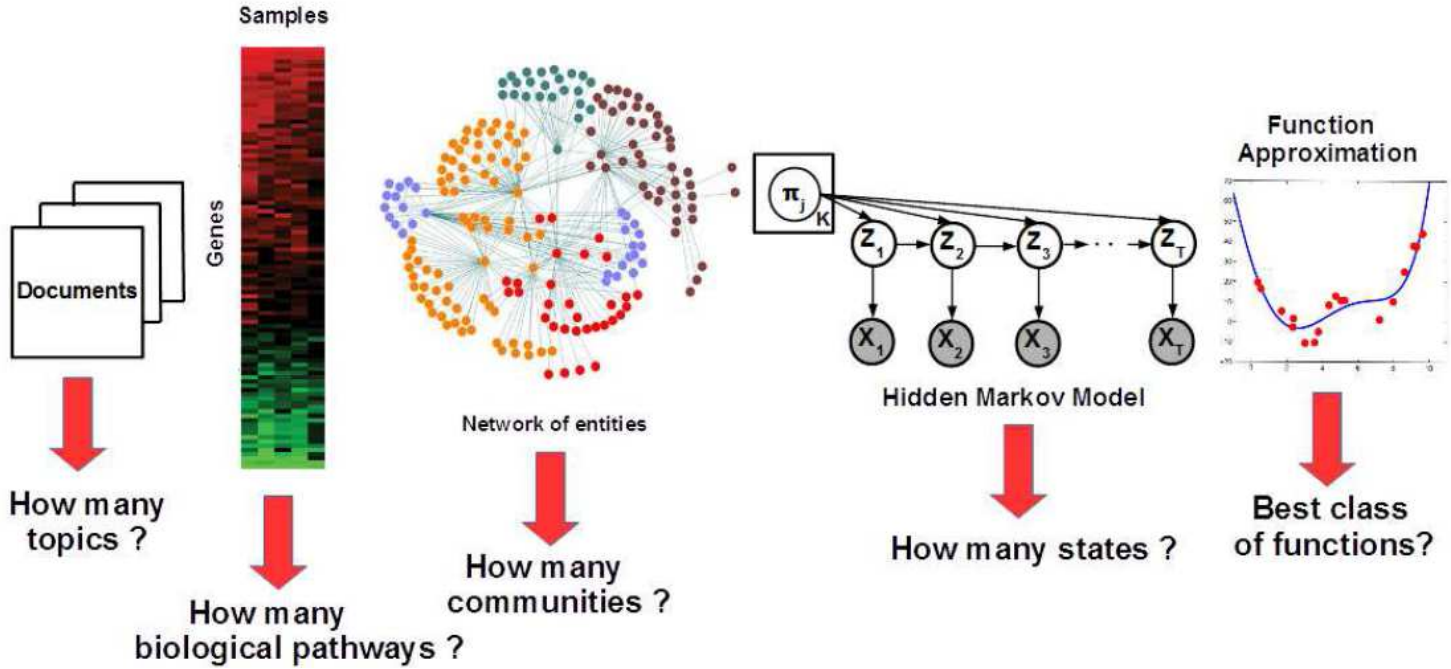
Nonparametric Bayesian Learning

- How big/complex my model should be? How many parameters suffice?



Nonparametric Bayesian Learning

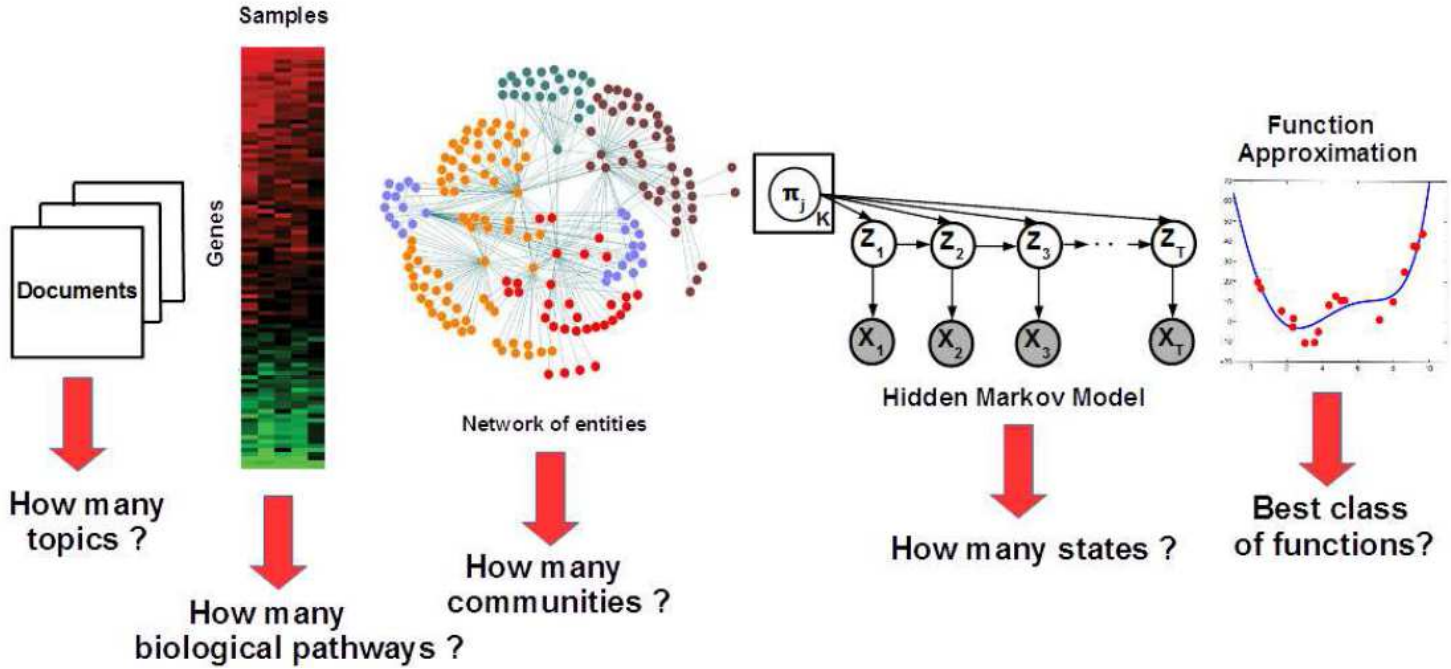
- How big/complex my model should be? How many parameters suffice?



- Model-selection or cross-validation, can often be expensive and impractical

Nonparametric Bayesian Learning

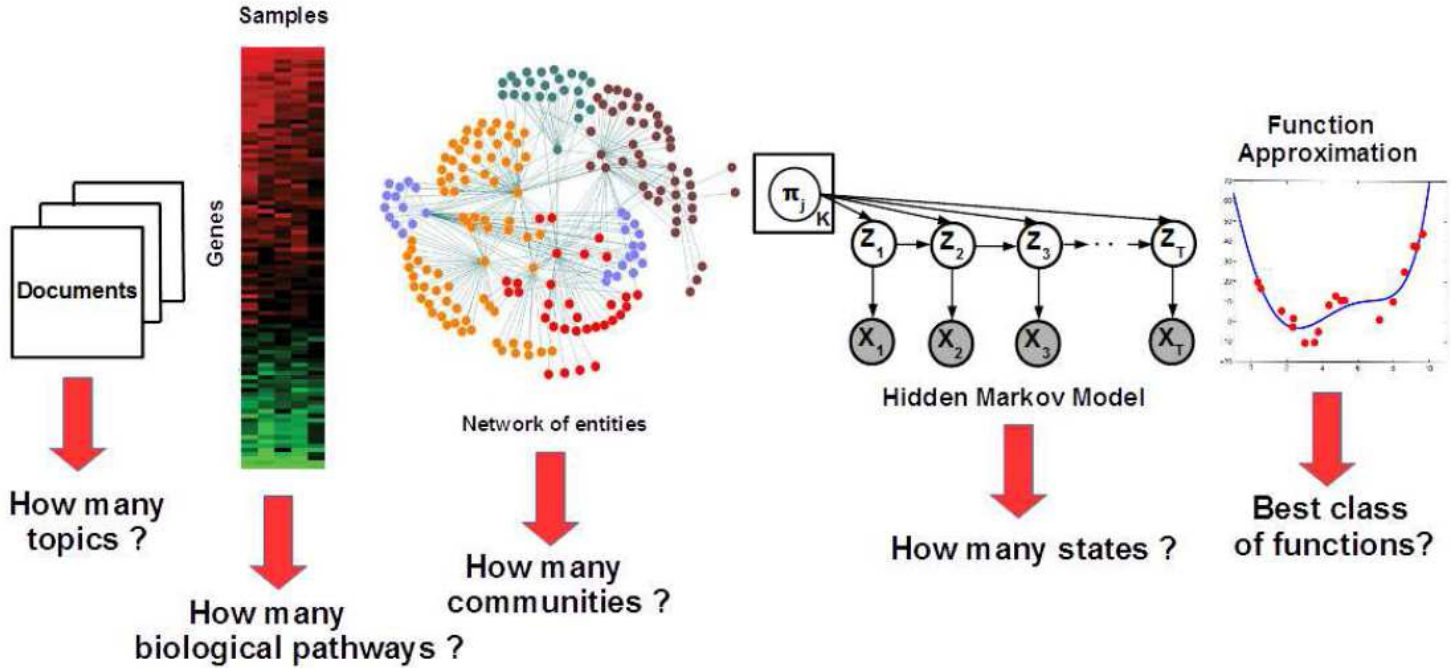
- How big/complex my model should be? How many parameters suffice?



- Model-selection or cross-validation, can often be expensive and impractical
- Nonparametric Bayesian Models: Allow **unbounded number** of parameters

Nonparametric Bayesian Learning

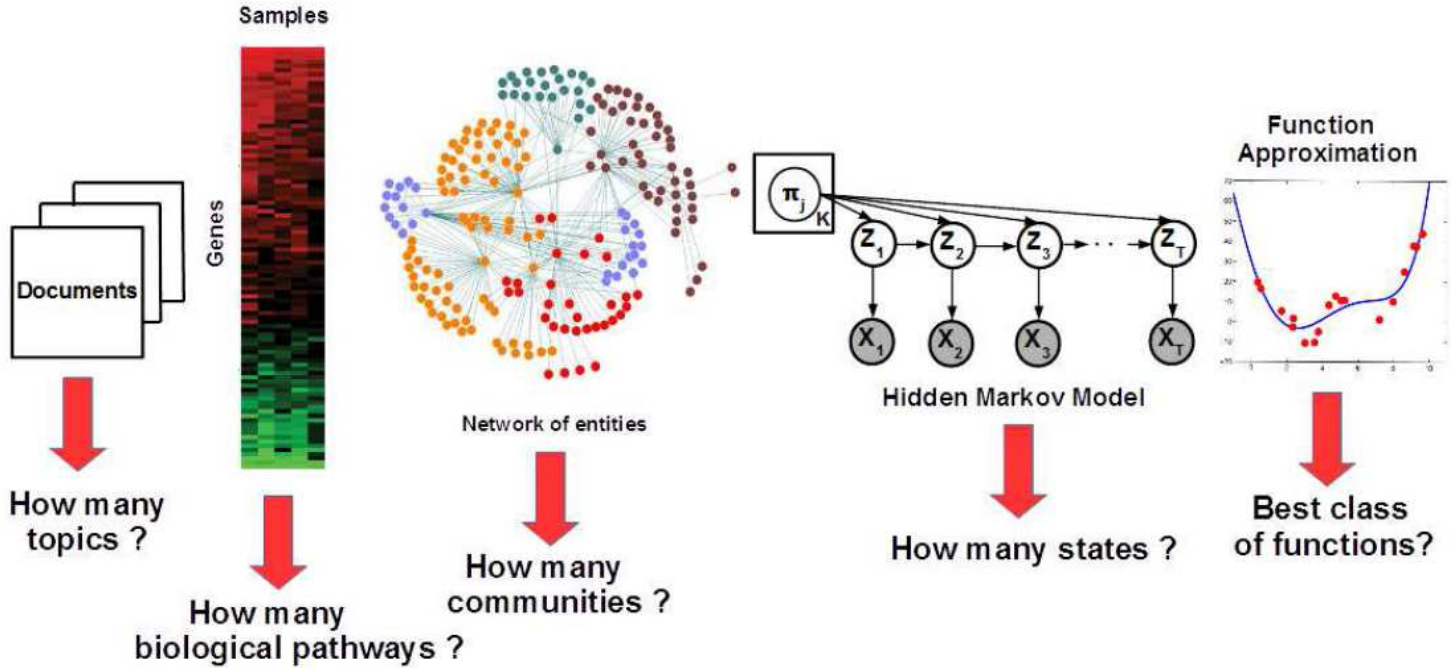
- How big/complex my model should be? How many parameters suffice?



- Model-selection or cross-validation, can often be expensive and impractical
- Nonparametric Bayesian Models: Allow **unbounded number** of parameters
  - The model can **grow/shrink adaptively** as we observe more and more data

Nonparametric Bayesian Learning

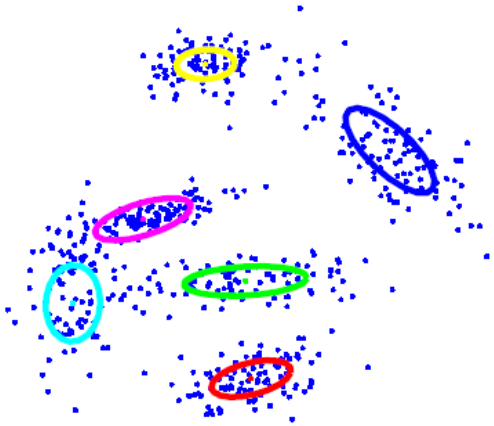
- How big/complex my model should be? How many parameters suffice?

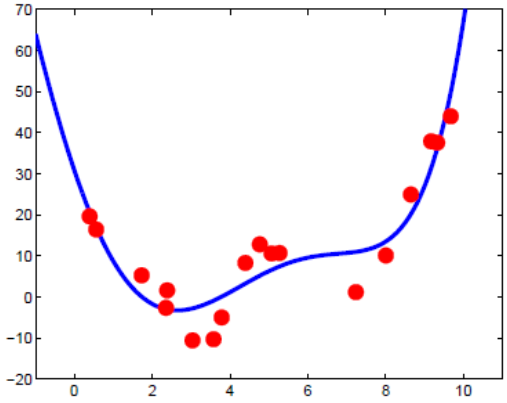


- Model-selection or cross-validation, can often be expensive and impractical
- Nonparametric Bayesian Models: Allow **unbounded number** of parameters
  - The model can **grow/shrink adaptively** as we observe more and more data
  - We "let the data speak" how complex the model needs to be

What's a Nonparametric Bayesian Model?

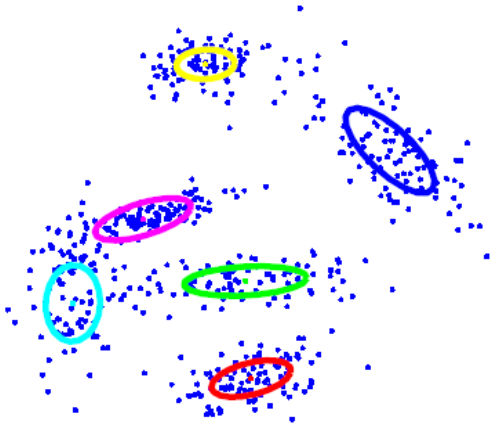
- An NPBayes model is NOT a model with no parameters!
- It has potentially infinite many (unbounded number of) parameters
- It has the ability to "create" new parameters if data requires so..

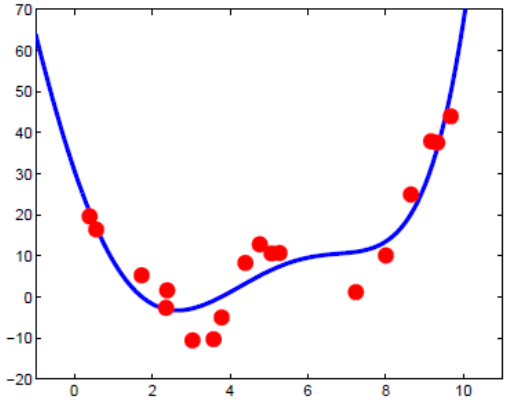




What's a Nonparametric Bayesian Model?

- An NPBayes model is NOT a model with no parameters!
- It has potentially infinite many (unbounded number of) parameters
- It has the ability to "create" new parameters if data requires so..

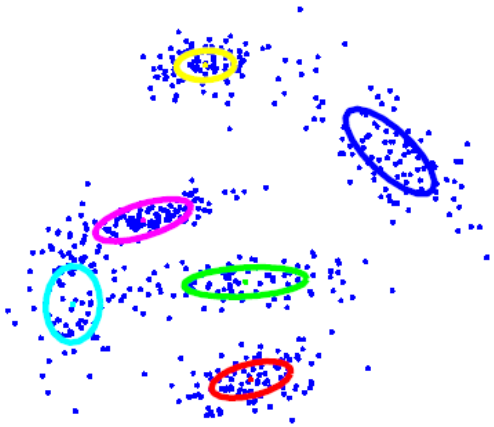


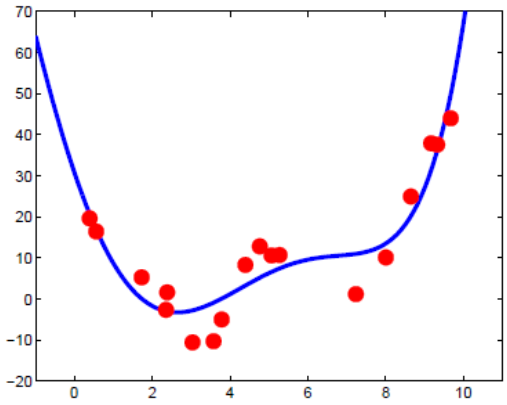


- Some non-Bayesian models are also nonparametric. For example: nearest neighbor regression/classification, kernel SVMs, kernel density estimation

What's a Nonparametric Bayesian Model?

- An NPBayes model is NOT a model with no parameters!
- It has potentially infinite many (unbounded number of) parameters
- It has the ability to "create" new parameters if data requires so..





- Some non-Bayesian models are also nonparametric. For example: nearest neighbor regression/classification, kernel SVMs, kernel density estimation
- NPBayes models offer the benefits of both Bayesian modeling and nonparametric modeling

Examples of NPBayes Models

Some modeling problems and NPBayes models of choice[1]:

Distributions on functions	Gaussian process
Distributions on distributions	Dirichlet process
	Polya Tree
Clustering	Chinese restaurant process
	Pitman-Yor process
Hierarchical clustering	Dirichlet diffusion tree
	Kingman's coalescent
Sparse binary matrices	Indian buffet processes
Survival analysis	Beta processes
Distributions on measures	Completely random measures
...	...

[1] Table courtesy: Zoubin Ghahramani

Gaussian Process

- A Gaussian Process (GP) is a distribution over functions f: $f \sim GP(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- .. such that f's value at a finite set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is jointly Gaussian

$$\{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

Gaussian Process

- A Gaussian Process (GP) is a distribution over functions f: $f \sim GP(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- .. such that f's value at a finite set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is jointly Gaussian

$$\{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- If $\boldsymbol{\mu} = \mathbf{0}$, a GP is fully specified by its covariance (kernel) matrix $\mathbf{K}$

Gaussian Process

- A Gaussian Process (GP) is a distribution over functions f: $f \sim GP(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- .. such that f's value at a finite set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is jointly Gaussian

$$\{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- If $\boldsymbol{\mu} = \mathbf{0}$, a GP is fully specified by its covariance (kernel) matrix $\mathbf{K}$
- Covariance matrix defined by a kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Some examples:
  - $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{2\sigma^2}\right)$: Gaussian kernel
  - $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = v_0 \exp\left\{-\left(\frac{|\boldsymbol{x}_n - \boldsymbol{x}_m|}{r}\right)^{\alpha}\right\} + v_1 + v_2 \delta_{nm}$

Gaussian Process

- A Gaussian Process (GP) is a distribution over functions f: $f \sim GP(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- .. such that f's value at a finite set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is jointly Gaussian

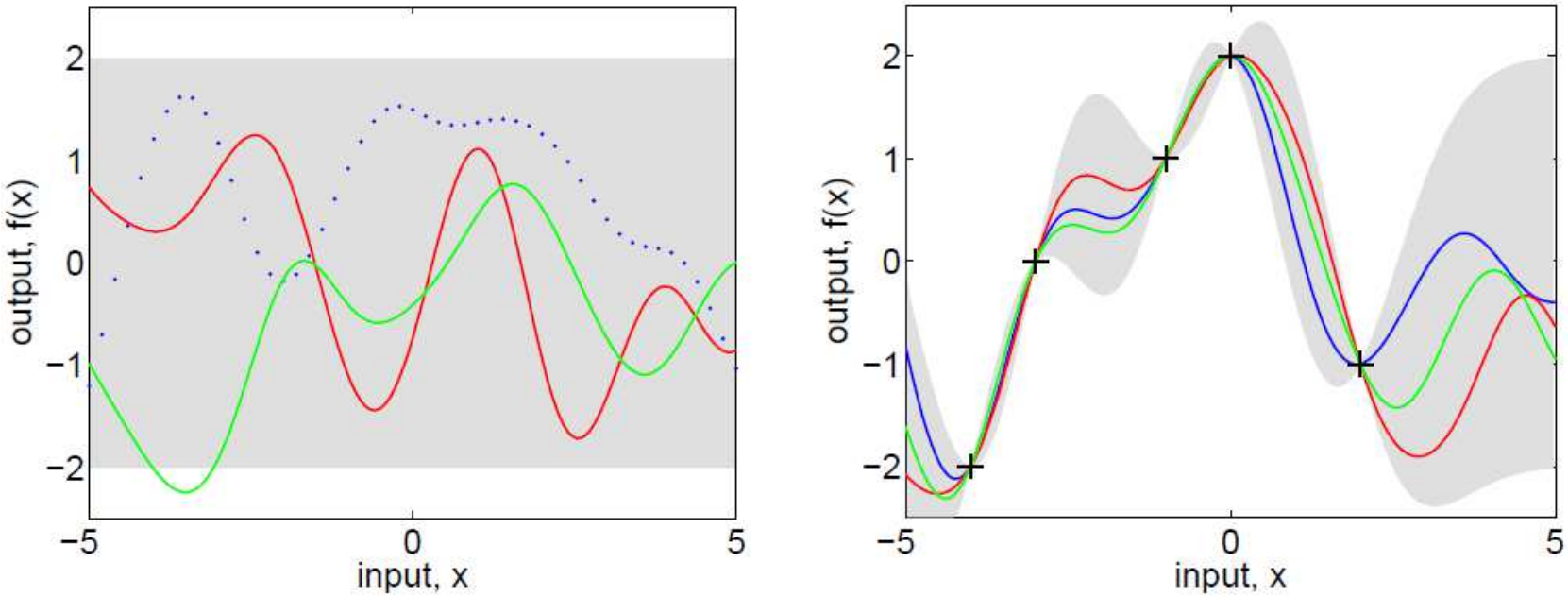
$$\{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- If $\boldsymbol{\mu} = \mathbf{0}$, a GP is fully specified by its covariance (kernel) matrix $\mathbf{K}$
- Covariance matrix defined by a kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Some examples:
  - $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{2\sigma^2}\right)$: Gaussian kernel
  - $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = v_0 \exp\left\{-\left(\frac{|\boldsymbol{x}_n - \boldsymbol{x}_m|}{r}\right)^{\alpha}\right\} + v_1 + v_2 \delta_{nm}$
- GP based modeling also allows learning the kernel hyperparameters from data

Gaussian Process

Left: some functions drawn from a GP prior $\mathcal{N}(\mathbf{0}, \mathbf{K})$

Right: posterior over these functions after observing 5 examples $\{\boldsymbol{x}_n, y_n\}$



Gaussian Process Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. Response is a noisy function of the input

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n$$

- Assume a zero-mean Gaussian error

$$p(\epsilon|\sigma^2) = \mathcal{N}(\epsilon|0, \sigma^2)$$

- Leads to a Gaussian likelihood model for the responses

$$p(y_n|f(\boldsymbol{x}_n)) = \mathcal{N}(y_n|f(\boldsymbol{x}_n), \sigma^2)$$

Gaussian Process Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. Response is a noisy function of the input

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n$$

- Assume a zero-mean Gaussian error

$$p(\epsilon|\sigma^2) = \mathcal{N}(\epsilon|0, \sigma^2)$$

- Leads to a Gaussian likelihood model for the responses

$$p(y_n|f(\boldsymbol{x}_n)) = \mathcal{N}(y_n|f(\boldsymbol{x}_n), \sigma^2)$$

- Denote $\boldsymbol{y} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$, $\mathbf{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$ and write

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

Gaussian Process Regression

- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. Response is a noisy function of the input

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n$$

- Assume a zero-mean Gaussian error

$$p(\epsilon|\sigma^2) = \mathcal{N}(\epsilon|0, \sigma^2)$$

- Leads to a Gaussian likelihood model for the responses

$$p(y_n|f(\boldsymbol{x}_n)) = \mathcal{N}(y_n|f(\boldsymbol{x}_n), \sigma^2)$$

- Denote $\boldsymbol{y} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$, $\mathbf{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$ and write

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- In GP regression, we assume f drawn from a GP

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

Gaussian Process Regression

- The likelihood model

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2\mathbf{I}_N)$$

- The prior distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The marginal distribution over the responses $\boldsymbol{y}$

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2\mathbf{I}_N + \mathbf{K})$$

Gaussian Process Regression

- The likelihood model

$$p(\boldsymbol{y}|\mathbf{f}) = \mathcal{N}(\boldsymbol{y}|\mathbf{f}, \sigma^2 \mathbf{I}_N)$$

- The prior distribution

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The marginal distribution over the responses $\boldsymbol{y}$

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2 \mathbf{I}_N + \mathbf{K})$$

Marginal and Conditional of Gaussians

If, $p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})$, and $p(y|x) = \mathcal{N}(y|Ax + b, L^{-1})$, then the marginal, $p(y)$, and the conditional $p(x|y)$ distributions are also Gaussians and are given by,

$$p(y) = \mathcal{N}(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^\top),$$
$$p(x|y) = \mathcal{N}(x|\Sigma(A^\top L(y - b) + \Lambda\mu), \Sigma).$$

Making Predictions

- Recall, the marginal distribution over the responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2\mathbf{I}_N + \mathbf{K}) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \mathbf{C}_N)$$

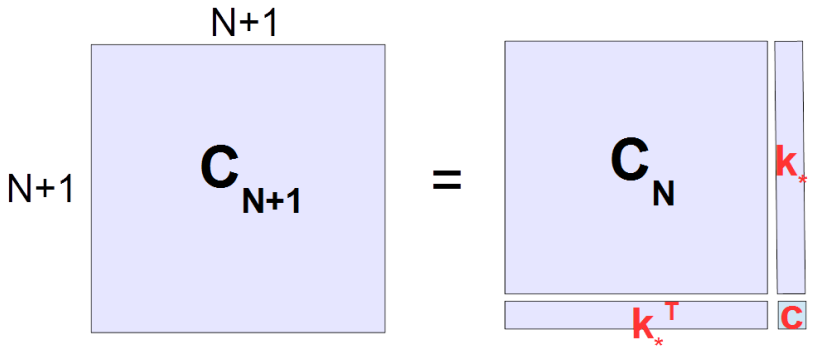
- Adding the response y_* of a new test point $\boldsymbol{x}_*$

$$p([\boldsymbol{y}, y_*]) = \mathcal{N}([\boldsymbol{y}, y_*]|\mathbf{0}, \mathbf{C}_{N+1})$$

where the $(N+1) \times (N+1)$ matrix $\mathbf{C}_{N+1}$ is given by

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C} & \mathbf{k}_* \\ \mathbf{k}_*^\top & c \end{bmatrix}$$

and $\mathbf{k}_* = [k(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_*, \boldsymbol{x}_N)]$, $c = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2$



Making Predictions on Test Data

- Recall $p([\boldsymbol{y}, y_*]) = \mathcal{N}([\boldsymbol{y}, y_*]|\mathbf{0}, \mathbf{C}_{N+1})$. The predictive distribution will be

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \frac{p([\boldsymbol{y}, y_*])}{p(\boldsymbol{y})} \\ p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|m(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)) \\ m(\boldsymbol{x}_*) &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma^2(\boldsymbol{x}_*) &= c - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

Making Predictions on Test Data

- Recall $p([\boldsymbol{y}, y_*]) = \mathcal{N}([\boldsymbol{y}, y_*]|\mathbf{0}, \mathbf{C}_{N+1})$. The predictive distribution will be

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \frac{p([\boldsymbol{y}, y_*])}{p(\boldsymbol{y})} \\ p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|m(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)) \\ m(\boldsymbol{x}_*) &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma^2(\boldsymbol{x}_*) &= c - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

Partitioned Gaussians

If x_a and x_b are Gaussian variables respectively with means μ_a and μ_b, then the conditional distribution, $p(x_a|x_b)$ is also Gaussian with mean and variance respectively, $\mu_{a|b}$ and $\Sigma_{a|b}$, given by,

$$\begin{aligned} \mu_{a|b} &= \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b), \\ \Sigma_{a|b} &= \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}. \end{aligned}$$

Making Predictions on Test Data

- Recall $p([\boldsymbol{y}, y_*]) = \mathcal{N}([\boldsymbol{y}, y_*]|\mathbf{0}, \mathbf{C}_{N+1})$. The predictive distribution will be

$$
\begin{aligned}
p(y_*|\boldsymbol{y}) &= \frac{p([\boldsymbol{y}, y_*])}{p(\boldsymbol{y})} \\
p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|m(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)) \\
m(\boldsymbol{x}_*) &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\
\sigma^2(\boldsymbol{x}_*) &= c - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \mathbf{k}_*
\end{aligned}
$$

Partitioned Gaussians

If x_a and x_b are Gaussian variables respectively with means μ_a and μ_b, then the conditional distribution, $p(x_a|x_b)$ is also Gaussian with mean and variance respectively, $\mu_{a|b}$ and $\Sigma_{a|b}$, given by,

$$
\begin{aligned}
\mu_{a|b} &= \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b), \\
\Sigma_{a|b} &= \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}.
\end{aligned}
$$

- Note that for GP regression, **exact inference** is possible at test time!

Interpreting GP predictions..

- Let's look at the predictions made by GP regression

$$\begin{aligned} p(y_*|\boldsymbol{y}) &= \mathcal{N}(y_*|m(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)) \\ m(\boldsymbol{x}_*) &= \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} \\ \sigma^2(\boldsymbol{x}_*) &= c - \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \mathbf{k}_* \end{aligned}$$

- Two interpretations for the mean prediction $m(\boldsymbol{x}_*)$
  - An SVM like interpretation

$$m(\boldsymbol{x}_*) = \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} = \mathbf{k}_*{}^\top \boldsymbol{\alpha} = \sum_{n=1}^{N} k(\boldsymbol{x}_*, \boldsymbol{x}_n)\alpha_n$$

    where $\boldsymbol{\alpha}$ is akin to the weights of support vectors

  - A nearest neighbors interpretation

$$m(\boldsymbol{x}_*) = \mathbf{k}_*{}^\top \mathbf{C}_N^{-1} \boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{y} = \sum_{n=1}^{N} w_n y_n$$

    where $\boldsymbol{w}$ is akin to the weights of the neighbors

Inferring Hyperparameters

- Recall, the marginal distribution over the responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

$$p(\boldsymbol{y}|\sigma^2, \theta) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \sigma^2\mathbf{I}_N + \mathbf{K}_\theta)$$

- Can maximize the (log) marginal likelihood w.r.t. σ^2 and the kernel hyperparameterss θ and get point estimates of the hyperparameters

$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

Inferring Hyperparameters

- Recall, the marginal distribution over the responses $\boldsymbol{y} = [y_1, \ldots, y_N]$

$$p(\boldsymbol{y}|\sigma^2, \theta) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2\mathbf{I}_N + \mathbf{K}_\theta)$$

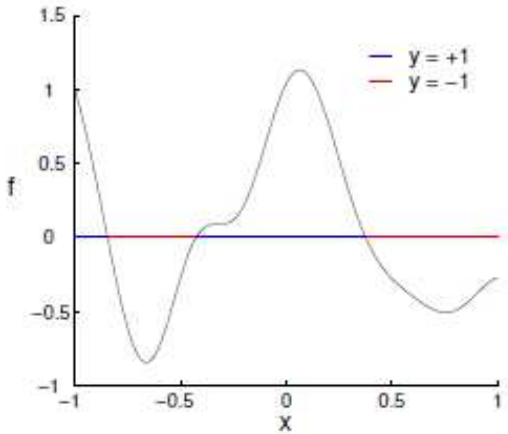
- Can maximize the (log) marginal likelihood w.r.t. σ^2 and the kernel hyperparameterss θ and get point estimates of the hyperparameters

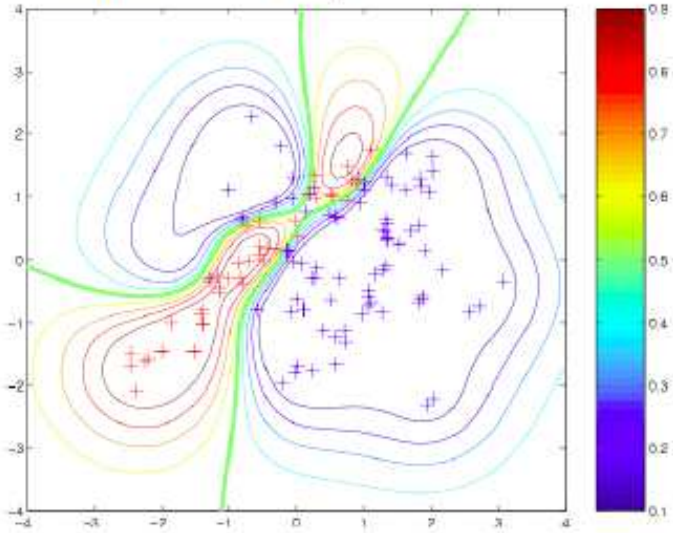
$$\log p(\boldsymbol{y}|\sigma^2, \theta) = -\frac{1}{2}\log|\sigma^2\mathbf{I}_N + \mathbf{K}_\theta| - \frac{1}{2}\boldsymbol{y}^\top(\sigma^2\mathbf{I}_N + \mathbf{K}_\theta)^{-1}\boldsymbol{y} + \text{const}$$

- Note: Can also put hyperpriors on the hyperparameters and infer the hyperparameters in a fully Bayesian manner

Gaussian Process Classification

Binary classification problem: Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with binary class labels $y_i \in \{-1, +1\}$, infer class label probabilities at new points.





There are many ways to relate function values $f_i = f(\mathbf{x}_i)$ to class probabilities:

$$p(y_i|f_i) = \begin{cases} \frac{1}{1+\exp(-y_i f_i)} & \text{sigmoid (logistic)} \\ \Phi(y_i f_i) & \text{cumulative normal (probit)} \\ \boldsymbol{H}(y_i f_i) & \text{threshold} \\ \epsilon + (1-2\epsilon)\boldsymbol{H}(y_i f_i) & \text{robust threshold} \end{cases}$$

Non-Gaussian likelihood, so we need to use approximate inference methods (Laplace, EP, MCMC).

- Non-binary labels (multiclass, counts, etc.) can also be easily handled

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

  where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

- Kernel SVM: $f_n = \sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Denote $\mathbf{f} = [f_1, \ldots, f_N]^\top$

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

  where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

- Kernel SVM: $f_n = \sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Denote $\mathbf{f} = [f_1, \ldots, f_N]^\top$

- We can write $\frac{||\boldsymbol{w}||^2}{2} = \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} = \mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$, and kernel SVM objective becomes

$$\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

GP vs (Kernel) SVM

- The objective function of a soft-margin SVM looks like

$$\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

  where $f_n = \boldsymbol{w}^\top \boldsymbol{x}_n$ and y_n is the true label for $\boldsymbol{x}_n$

- Kernel SVM: $f_n = \sum_{m=1}^{N} \alpha_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Denote $\mathbf{f} = [f_1, \ldots, f_N]^\top$

- We can write $\frac{||\boldsymbol{w}||^2}{2} = \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} = \mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$, and kernel SVM objective becomes

$$\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_{n=1}^{N}(1 - y_n f_n)_+$$

- Negative log of the likelihood $p(\mathbf{f}|\mathbf{X})$ of a GP can be written as

$$\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} - \sum_{n=1}^{N} \log p(y_n|f_n) + \text{const}$$

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,
  - Learning the kernel bandwidth for Gaussian kernels

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{2\sigma^2}\right)$$

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,
  - Learning the kernel bandwidth for Gaussian kernels

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{2\sigma^2}\right)$$

  - Doing feature selection (via Automatic Relevance Determination)

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D}\frac{(\boldsymbol{x}_{nd} - \boldsymbol{x}_{md})^2}{2{\sigma_d}^2}\right)$$

GP vs (Kernel) SVM

- Thus GPs can be interpreted as a Bayesian analogue of kernel SVMs
- Both GP and SVM need dealing with (storing/inverting) large kernel matrices
  - Various approximations proposed to address this issue (applicable to both)
- Ability to learn the kernel hyperparameters in GP is very useful, e.g.,
  - Learning the kernel bandwidth for Gaussian kernels

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\frac{||\boldsymbol{x}_n - \boldsymbol{x}_m||^2}{2\sigma^2}\right)$$

  - Doing feature selection (via Automatic Relevance Determination)

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp\left(-\sum_{d=1}^{D}\frac{(\boldsymbol{x}_{nd} - \boldsymbol{x}_{md})^2}{2{\sigma_d}^2}\right)$$

  - Learning compositions of kernels for more flexible modeling

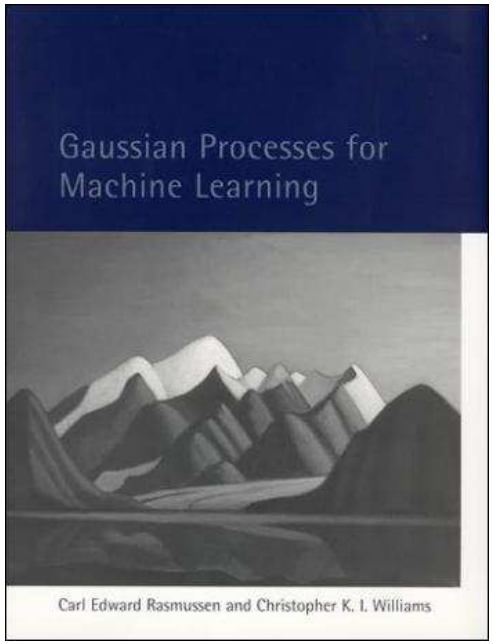
$$\mathbf{K} = \mathbf{K}_{\theta_1} + \mathbf{K}_{\theta_2} + \ldots$$

Other Usage of GP

- **Nonlinear Dimensionality Reduction:** Gaussian Process Latent Variable Models
- **Bayesian Optimization:** Optimizing functions that have an unknown functional form and are expensive to evaluate
- **Deep Gaussian Processes:** Data assumed to be an output of a multivariate GP, inputs to each GP are outputs of another GP, and so on..
- Many applications: Robotics and control, vision, spatial statistics, and so on..

Resources on Gaussian Processes

- Book: Gaussian Processes for Machine Learning (freely available online)



- MATLAB Packages: Useful to play with, build applications, extend existing models and inference algorithms for GPs (both regression and classification)
  - GPML: `http://www.gaussianprocess.org/gpml/code/matlab/doc/`
  - GPStuff: `http://research.cs.aalto.fi/pml/software/gpstuff/`

Next Talk

- Nonparametric Bayesian models for mixture modeling (clustering): Dirichlet Processes and Chinese Restaurant Process
- Nonparametric Bayesian models for latent factor modeling (dimensionality reduction): Beta Processes and Indian Buffet Process

Thanks! Questions?