From Linear Models to Gaussian Processes

Mark van der Wilk

Department of Computing
Imperial College London

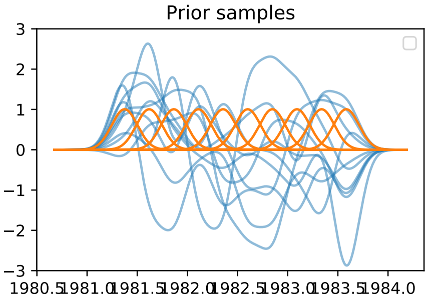
@markvanderwilk
m.vdwilk@imperial.ac.uk

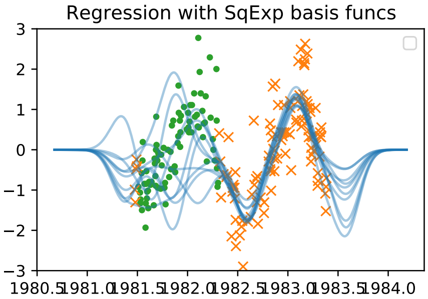
January 23, 2023

Recap

Last lecture we saw that:

- The prior has a large effect on the predictions & we needed a sensible prior.
- Polynomial bases didn't lead to a sensible prior, squared exponential did.
- We needed many basis functions to ensure sensible uncertainty.

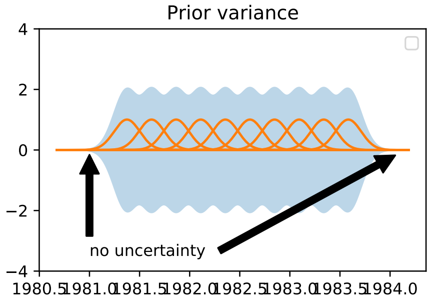


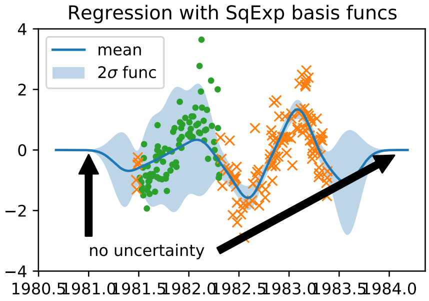


Recap

Last lecture we saw that:

- The prior has a large effect on the predictions & we needed a sensible prior.
- Polynomial bases didn't lead to a sensible prior, squared exponential did.
- We needed many basis functions to ensure sensible uncertainty.

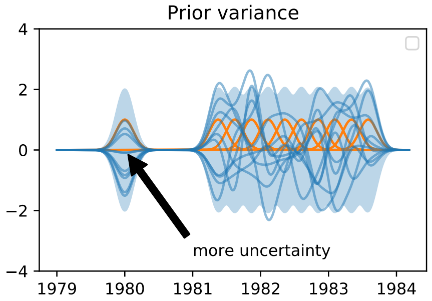


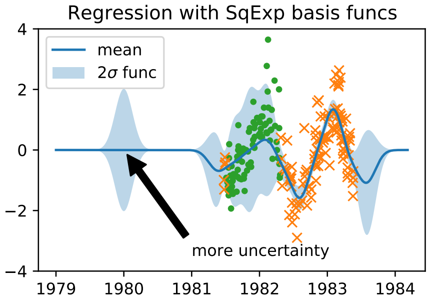


Recap

Last lecture we saw that:

- The prior has a large effect on the predictions & we needed a sensible prior.
- Polynomial bases didn't lead to a sensible prior, squared exponential did.
- We needed many basis functions to ensure sensible uncertainty.

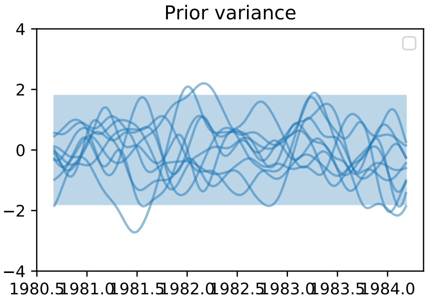


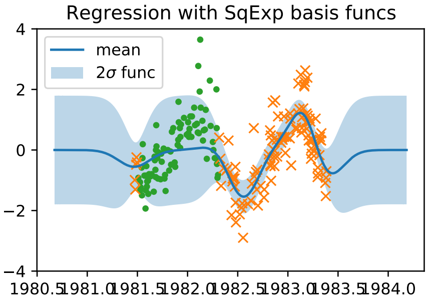


Recap

Last lecture we saw that:

- The prior has a large effect on the predictions & we needed a sensible prior.
- Polynomial bases didn't lead to a sensible prior, squared exponential did.
- We needed many basis functions to ensure sensible uncertainty.



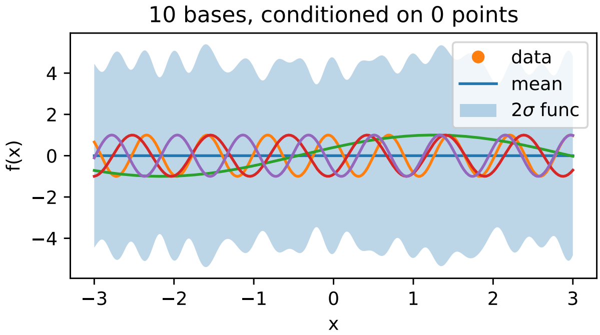


Today

We will see:

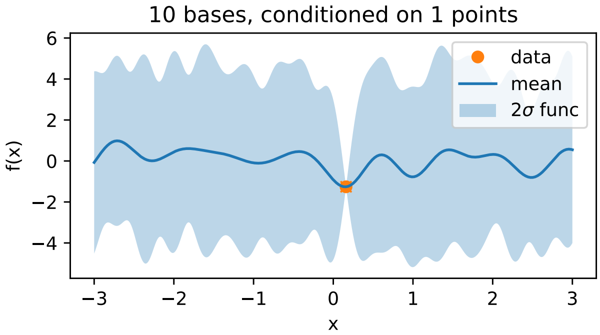
- By considering computational cost, we derive the Gaussian process view of BLR.
- This is the kernel trick!
- What is a Gaussian process.
- How to find posteriors in GP models.

Infinite Basis Functions: Another Reason



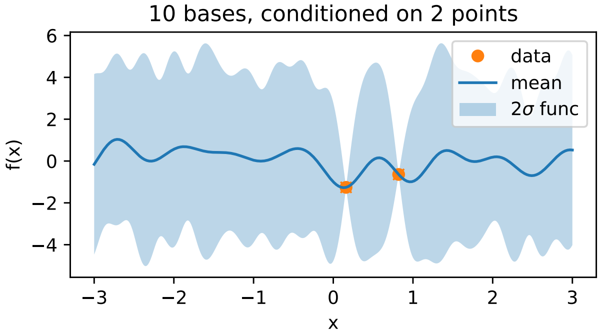
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



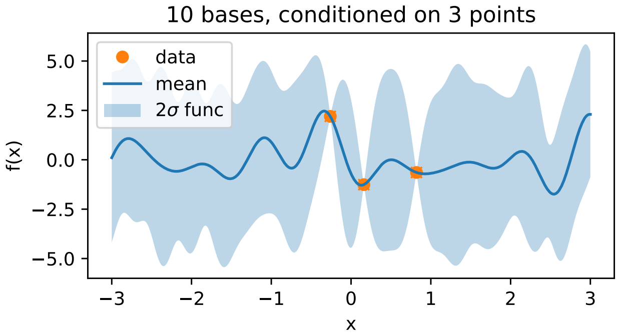
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



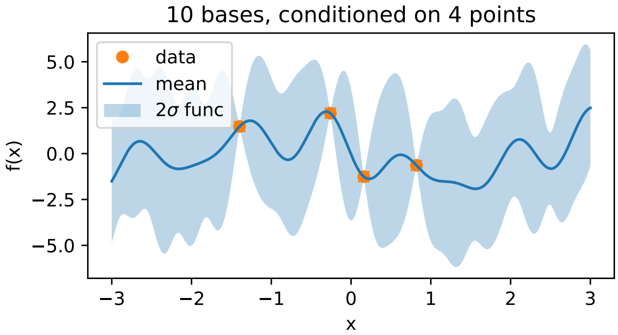
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



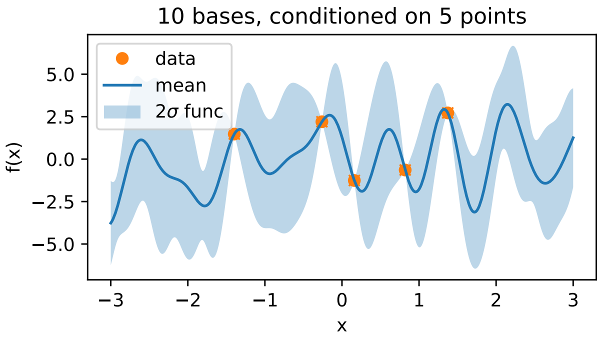
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



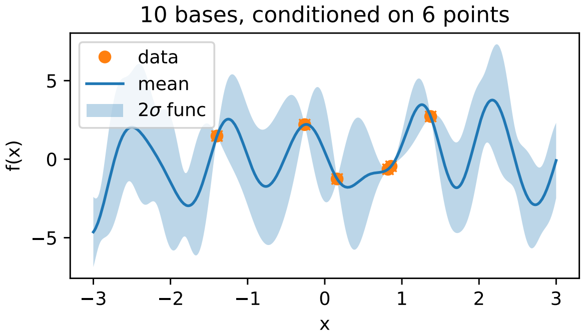
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



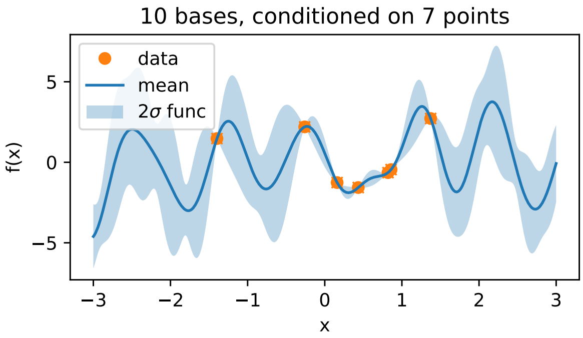
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



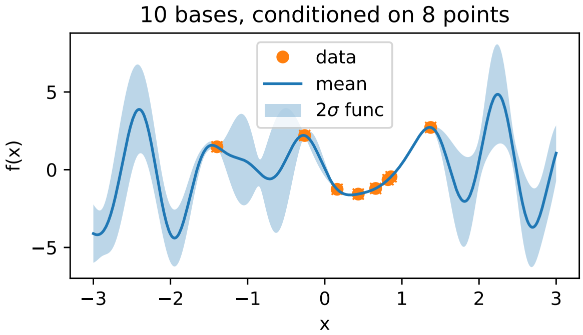
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



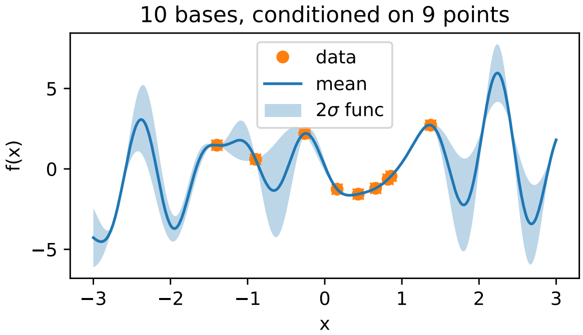
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



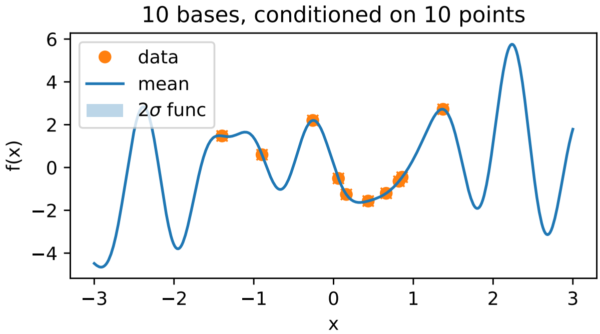
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



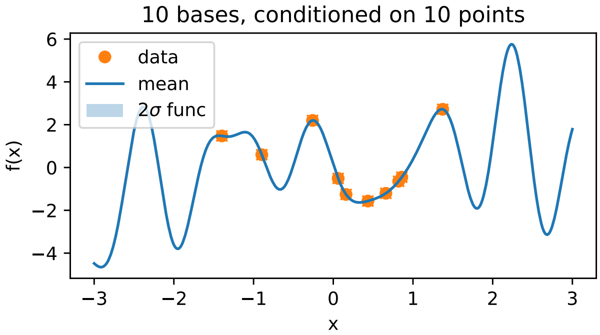
- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Infinite Basis Functions: Another Reason



- Non-local basis functions give uncertainty everywhere (like polynomials).
- But uncertainty decreases non-locally!
- For M bases, when we condition on M points: full certainty!

Exercise: For a model with M bases, show that after conditioning on M points (zero variance likelihood) leads to 1) completely certain predictions, 2) certain posterior.

BLR: Computing the Posterior

For Gaussian models, finding conditionals can easily be done by finding the **joint**, and then applying the **Gaussian conditioning rule**.

$$\boldsymbol{\theta} \sim \mathcal{N}(0, \boldsymbol{I}_M)\,, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \boldsymbol{I}_N\sigma^2\right), \qquad [\Phi(\boldsymbol{X})]_{nm} = \phi_m(\mathbf{x}_n)\,. \tag{1}$$

$$\boldsymbol{\theta} \in \mathbb{R}^M, \quad \mathbf{y} \in \mathbb{R}^N, \quad \boldsymbol{\epsilon} \in \mathbb{R}^N, \quad \Phi(\boldsymbol{X}) \in \mathbb{R}^{N\times M}, \quad \boldsymbol{X} \in \mathbb{R}^{N\times D}\,. \tag{2}$$

$$\begin{bmatrix}\boldsymbol{\theta}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}_M & 0\\ \Phi(\boldsymbol{X}) & \boldsymbol{I}_n\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}\\ \boldsymbol{\epsilon}\end{bmatrix} \tag{3}$$

$$\implies p\left(\begin{bmatrix}\boldsymbol{\theta}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\theta}\\ \mathbf{y}\end{bmatrix}; 0, \begin{bmatrix}\boldsymbol{I}_M & \Phi(\boldsymbol{X})^\intercal\\ \Phi(\boldsymbol{X}) & \Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\end{bmatrix}\right) \tag{4}$$

Using:

- Linear relationships between Gaussian RVs gives Gaussian joint.
  - Write joint Gaussian as a linear transformation of RVs with known independent distributions.
- $\mathbb{E}_{\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma})}[\boldsymbol{A}\mathbf{x}] = \boldsymbol{A}\boldsymbol{\mu}$, and $\mathbb{V}_{\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma})}[\boldsymbol{A}\mathbf{x}] = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\intercal$.

BLR: Computing the Posterior

Gaussian conditioning formula (will be provided in exam):

$$p\left(\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right) \tag{5}$$

$$p(\mathbf{x}_2|\mathbf{x}_1) = \mathcal{N}\left(\mathbf{x}_2; \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\Sigma_{11}^{-1}(\mathbf{x}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right) \tag{6}$$

$p(\mathbf{x}_1|\mathbf{x}_2)$ is similar, and can be obtained by reordering the vector to $\begin{bmatrix}\mathbf{x}_2 & \mathbf{x}_1\end{bmatrix}^\intercal$. You can find the covariance matrix for this ordering in terms of the covariance blocks that are given above.

BLR: Computing the Posterior

Gaussian conditioning formula (will be provided in exam):

$$p\left(\begin{bmatrix}\mathbf{x}_1\\\mathbf{x}_2\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1\\\mathbf{x}_2\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_1\\\boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\\boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right) \tag{5}$$

$$p(\mathbf{x}_2|\mathbf{x}_1) = \mathcal{N}\left(\mathbf{x}_2; \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\Sigma_{11}^{-1}(\mathbf{x}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right) \tag{6}$$

$p(\mathbf{x}_1|\mathbf{x}_2)$ is similar, and can be obtained by reordering the vector to $\begin{bmatrix}\mathbf{x}_2 & \mathbf{x}_1\end{bmatrix}^\intercal$. You can find the covariance matrix for this ordering in terms of the covariance blocks that are given above.

$$p\left(\begin{bmatrix}\boldsymbol{\theta}\\\mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\theta}\\\mathbf{y}\end{bmatrix}; 0, \begin{bmatrix}\boldsymbol{I}_M & \Phi(\boldsymbol{X})^\intercal\\\Phi(\boldsymbol{X}) & \Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\end{bmatrix}\right) \tag{7}$$

$$p(\boldsymbol{\theta}|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \Phi(\boldsymbol{X})^\intercal\left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\right]^{-1}\mathbf{y}$$

$$\mathbf{I}_M - \Phi(\boldsymbol{X})^\intercal\left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\right]^{-1}\Phi(\boldsymbol{X})\Big) \tag{8}$$

BLR: Computing the Posterior

Looks complicated. But we can compute it!

$$p(\boldsymbol{\theta}|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \Phi(\boldsymbol{X})^\top\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top + \sigma^2\boldsymbol{I}_N\big]^{-1}\mathbf{y} \\ \mathbf{I}_M - \Phi(\boldsymbol{X})^\top\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top + \sigma^2\boldsymbol{I}_N\big]^{-1}\Phi(\boldsymbol{X})\Big) \tag{9}$$

What is the computational cost?

[1] $N \times N$ matrix multiplication and matrix inversion can both be $O(N^{2.373})$, but we assume $O(N^3)$. Most important is that we distinguish these expensive operations from cheaper ones that are $O(N^2)$.

BLR: Computing the Posterior

Looks complicated. But we can compute it!

$$p(\boldsymbol{\theta}|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\big]^{-1}\mathbf{y}$$
$$\mathbf{I}_M - \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\big]^{-1}\Phi(\boldsymbol{X})\Big) \tag{9}$$

What is the computational cost? We assume costs of simple linear algebra algorithms, even though more efficient algorithms exist[1].

- $\Phi(\boldsymbol{X})$: $O(NMD)$ — Assume linear time cost for each dimension of input, then need to compute each basis function for each data point.

[1] $N \times N$ matrix multiplication and matrix inversion can both be $O(N^{2.373})$, but we assume $O(N^3)$. Most important is that we distinguish these expensive operations from cheaper ones that are $O(N^2)$.

BLR: Computing the Posterior

Looks complicated. But we can compute it!

$$p(\boldsymbol{\theta}|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\big]^{-1}\mathbf{y} \\ \mathbf{I}_M - \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\big]^{-1}\Phi(\boldsymbol{X})\Big) \tag{9}$$

What is the computational cost? We assume costs of simple linear algebra algorithms, even though more efficient algorithms exist[1].

- $\Phi(\boldsymbol{X})$: $O(NMD)$ — Assume linear time cost for each dimension of input, then need to compute each basis function for each data point.
- $\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal$: $O(N^2M)$ — Matrix multiplication

[1] $N \times N$ matrix multiplication and matrix inversion can both be $O(N^{2.373})$, but we assume $O(N^3)$. Most important is that we distinguish these expensive operations from cheaper ones that are $O(N^2)$.

BLR: Computing the Posterior

Looks complicated. But we can compute it!

$$p(\boldsymbol{\theta}|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\big]^{-1}\mathbf{y} \\ \mathbf{I}_M - \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\big]^{-1}\Phi(\boldsymbol{X})\Big) \tag{9}$$

What is the computational cost? We assume costs of simple linear algebra algorithms, even though more efficient algorithms exist[1].

- $\Phi(\boldsymbol{X})$: $O(NMD)$ — Assume linear time cost for each dimension of input, then need to compute each basis function for each data point.
- $\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal$: $O(N^2M)$ — Matrix multiplication
- $\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\big]^{-1}$ — $O(N^3)$ Matrix inversion (or Cholesky)

[1] $N \times N$ matrix multiplication and matrix inversion can both be $O(N^{2.373})$, but we assume $O(N^3)$. Most important is that we distinguish these expensive operations from cheaper ones that are $O(N^2)$.

Woodbury Identity (exam skill)

Usually $M \ll N$, so bottleneck: $\left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\right]^{-1}$ — $O(N^3)$

- Annoying that we have to compute an $O(N^3)$ cost inverse when the matrix we want is only $\mathbb{R}^{M\times M}$.
- Also, note that $\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal$ is at most rank M! **Low rank** matrices are usually cheaper to deal with!

[2] Matrix cookbook recipe 156

Woodbury Identity (exam skill)

Usually $M \ll N$, so bottleneck: $\left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top + \sigma^2 \boldsymbol{I}_N\right]^{-1}$ — $O(N^3)$

- Annoying that we have to compute an $O(N^3)$ cost inverse when the matrix we want is only $\mathbb{R}^{M\times M}$.
- Also, note that $\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top$ is at most rank M! **Low rank** matrices are usually cheaper to deal with!

Woodbury Identity[2]:

$$\underbrace{(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{B}\boldsymbol{V})^{-1}}_{N\times N} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}\underbrace{\left(\boldsymbol{B}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}\right)^{-1}}_{M\times M}\boldsymbol{V}\boldsymbol{A}^{-1} \tag{10}$$

$$\boldsymbol{A} \in \mathbb{R}^{N\times N}, \quad \boldsymbol{U} \in \mathbb{R}^{N\times M}, \quad \boldsymbol{V} \in \mathbb{R}^{M\times N}, \quad \boldsymbol{B} \in \mathbb{R}^{M\times M} \tag{11}$$

[2] Matrix cookbook recipe 156

BLR: Cheap Posterior Mean

Let's start with the mean:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} = \Phi(\boldsymbol{X})^\intercal \left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\right]^{-1} \mathbf{y} \tag{12}$$

and take $\boldsymbol{A} = \sigma^2 \boldsymbol{I}_N$, $\boldsymbol{U} = \Phi(\boldsymbol{X})$, $\boldsymbol{B} = \boldsymbol{I}_M$, $\boldsymbol{V} = \Phi(\boldsymbol{X})^\intercal$:

$$\left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\right]^{-1} = \frac{\boldsymbol{I}_N}{\sigma^2} - \frac{\Phi(\boldsymbol{X})}{\sigma^2}\left[\boldsymbol{I}_M + \frac{\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})}{\sigma^2}\right]^{-1}\frac{\Phi(\boldsymbol{X})}{\sigma^2}^\intercal$$

$$\begin{aligned}
\therefore \boldsymbol{\mu}_{\boldsymbol{\theta}} &= \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\left[\boldsymbol{I}_N - \frac{\Phi(\boldsymbol{X})}{\sigma^2}\left[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\right]^{-1}\Phi(\boldsymbol{X})^\intercal\right]\mathbf{y} \\
&= \left[\boldsymbol{I}_M - \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\left[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\right]^{-1}\right]\sigma^{-2}\Phi(\boldsymbol{X})^\intercal\mathbf{y} \\
&= \left[\left[\boldsymbol{I}_M + \cancel{\sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})}\right] - \cancel{\sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})}\right]\cdot \\
&\qquad \left[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\right]^{-1}\sigma^{-2}\Phi(\boldsymbol{X})^\intercal\mathbf{y}
\end{aligned} \tag{13}$$

BLR: Cheap Posterior Mean

$$\boldsymbol{\mu}_{\boldsymbol{\theta}} = \left[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\right]^{-1}\sigma^{-2}\Phi(\boldsymbol{X})^\intercal\mathbf{y} \tag{14}$$

Now we can compute in:

- $\Phi(\boldsymbol{X})$: $O(NMD)$ — As earlier.
- $\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})$: $O(M^2N)$ — Matrix multiplication
- $[\boldsymbol{I}_M + \Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})]^{-1}$ — $O(M^3)$ Matrix inversion (or Cholesky)

So when $M \ll N$, we now have $O(NM^2)$.

BLR: Cheap Posterior Variance

We can similarly apply Woodbury to the posterior variance, just slightly differently.
Always **remember the goal**! From large inverse, to small inverse.

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \mathbf{I}_M - \Phi(\boldsymbol{X})^\intercal \left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\right]^{-1} \Phi(\boldsymbol{X}) \tag{15}$$

We take $\boldsymbol{A}^{-1} = \boldsymbol{I}_M$, $\boldsymbol{U} = \Phi(\boldsymbol{X})^\intercal$, $\boldsymbol{B}^{-1} = \sigma^2 \boldsymbol{I}_M$, $\boldsymbol{V} = \Phi(\boldsymbol{X})^\intercal$ to obtain:

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \left[\boldsymbol{I}_M + \sigma^{-2} \Phi(\boldsymbol{X})^\intercal \Phi(\boldsymbol{X})\right]^{-1} \tag{16}$$

Also computable in $O(NM^2)$!

Two Ways to Compute

Method 1, cost $O(N^3 + N^2M + NMD)$:

$$p(\boldsymbol{\theta}|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\big]^{-1}\mathbf{y} \\ \mathbf{I}_M - \Phi(\boldsymbol{X})^\intercal\big[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2\boldsymbol{I}_N\big]^{-1}\Phi(\boldsymbol{X})\Big) \tag{17}$$

Method 2, cost $O(NM^2 + M^3 + NMD)$:

$$p(\boldsymbol{\theta}|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \big[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\big]^{-1}\sigma^{-2}\Phi(\boldsymbol{X})^\intercal\mathbf{y} \\ \big[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\intercal\Phi(\boldsymbol{X})\big]^{-1}\Big) \tag{18}$$

Predictive Distribution

Compute predictive distribution from mean and variance of $p(\boldsymbol{\theta}|\mathbf{y})$ was an exercise (`q&a_video_07` notes).

1. We find the posterior parameters in some way.
2. We apply Woodbury to ensure we take a small matrix inverse.
3. We get predictions at a cost of $O(NM^2 + M^3 + NMD)$.

Using the parameters found by method 2:

$$p(\mathbf{y}_*|\mathbf{y}) = \mathcal{N}\Big(\boldsymbol{\theta}; \quad \boldsymbol{\phi}(\mathbf{x}_*)^\top\left[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\top\Phi(\boldsymbol{X})\right]^{-1}\sigma^{-2}\Phi(\boldsymbol{X})^\top\mathbf{y} \\ \boldsymbol{\phi}(\mathbf{x}_*)^\top\left[\boldsymbol{I}_M + \sigma^{-2}\Phi(\boldsymbol{X})^\top\Phi(\boldsymbol{X})\right]^{-1}\boldsymbol{\phi}(\mathbf{x}_*) + \sigma^2\boldsymbol{I}_N\Big) \quad (19)$$

Predictive Distribution — Exercises

We can also find a different form of the predictive distribution, *without* finding the posterior over parameters first.

1. Using the method of transforming Gaussian RVs, show that the joint $p(\mathbf{y}, y_*)$ is

$$p(\mathbf{y}, y_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y} \\ y_*\end{bmatrix}; 0, \begin{bmatrix}\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N & \Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*) \\ \boldsymbol{\phi}(\mathbf{x}_*)^\intercal \Phi(\boldsymbol{X})^\intercal & \boldsymbol{\phi}(\mathbf{x}_*)^\intercal \boldsymbol{\phi}(\mathbf{x}_*) + \sigma^2\end{bmatrix}\right) \tag{20}$$

2. Show that

$$\begin{aligned} p(y_*|\mathbf{y}) = \mathcal{N}\Big(y_*; \quad & \boldsymbol{\phi}(\mathbf{x}_*)^\intercal \Phi(\boldsymbol{X})^\intercal \left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\right]^{-1} \mathbf{y}, \\ & \boldsymbol{\phi}(\mathbf{x}_*)^\intercal \boldsymbol{\phi}(\mathbf{x}_*) + \sigma^2 \\ & - \boldsymbol{\phi}(\mathbf{x}_*)^\intercal \Phi(\boldsymbol{X})^\intercal \left[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal + \sigma^2 \boldsymbol{I}_N\right]^{-1} \Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*)\Big) \end{aligned} \tag{21}$$

The cost of computing the predictive in this way is $O(N^3 + N^2 M + NMD)$ (like the earlier posterior).

Infinite Basis Functions

So we said that to *properly* model uncertainty, and have a flexible enough model, we needed *many*, or even an **infinite** number of basis functions.

- For the $O(NM^2 + M^3 + NMD)$ method, all terms contain $M \to \infty$ because each matrix we compute grows with the features.
- For the $O(N^3 + N^2M + NMD)$ method, the matrices we need are all of finite size...:

3

Infinite Basis Functions

So we said that to *properly* model uncertainty, and have a flexible enough model, we needed *many*, or even an **infinite** number of basis functions.

- For the $O(NM^2 + M^3 + NMD)$ method, all terms contain $M \to \infty$ because each matrix we compute grows with the features.
- For the $O(N^3 + N^2M + NMD)$ method, the matrices we need are all of finite size...:

$$\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\mathsf{T} \in \mathbb{R}^{N\times N}, \quad \Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*) \in \mathbb{R}^{N\times 1} \tag{22}$$

3

Infinite Basis Functions

So we said that to *properly* model uncertainty, and have a flexible enough model, we needed *many*, or even an **infinite** number of basis functions.

- For the $O(NM^2 + M^3 + NMD)$ method, all terms contain $M \to \infty$ because each matrix we compute grows with the features.
- For the $O(N^3 + N^2M + NMD)$ method, the matrices we need are all of finite size...:

$$\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top \in \mathbb{R}^{N\times N}, \quad \Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*) \in \mathbb{R}^{N\times 1} \tag{22}$$

Notice that we only need **inner products** between feature vectors:

$$[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top]_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)^\top\boldsymbol{\phi}(\mathbf{x}_j)\,. \tag{23}$$

3

Infinite Basis Functions

So we said that to *properly* model uncertainty, and have a flexible enough model, we needed *many*, or even an **infinite** number of basis functions.

- For the $O(NM^2 + M^3 + NMD)$ method, all terms contain $M \to \infty$ because each matrix we compute grows with the features.
- For the $O(N^3 + N^2M + NMD)$ method, the matrices we need are all of finite size...:

$$\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top \in \mathbb{R}^{N\times N}\,, \quad \Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*) \in \mathbb{R}^{N\times 1} \tag{22}$$

Notice that we only need **inner products** between feature vectors:

$$[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top]_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)^\top\boldsymbol{\phi}(\mathbf{x}_j)\,. \tag{23}$$

What if I told you... there were functions that computed inner products... without computing the vector itself?

3

Infinite Basis Functions

So we said that to *properly* model uncertainty, and have a flexible enough model, we needed *many*, or even an **infinite** number of basis functions.

- For the $O(NM^2 + M^3 + NMD)$ method, all terms contain $M \to \infty$ because each matrix we compute grows with the features.
- For the $O(N^3 + N^2M + NMD)$ method, the matrices we need are all of finite size...:

$$\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top \in \mathbb{R}^{N\times N}, \quad \Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*) \in \mathbb{R}^{N\times 1} \tag{22}$$

Notice that we only need **inner products** between feature vectors:

$$[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top]_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j)\,. \tag{23}$$

What if I told you... there were functions that computed inner products... without computing the vector itself? **Kernel trick.**[3]

[3] `http://oneweirdkerneltrick.com`

Kernels: Polynomial kernel

If we can compute the matrices $\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal$ and $\Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*)$ directly, without first computing the features, we could do computations without incurring the cost for large features!

Kernels: Polynomial kernel

If we can compute the matrices $\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal$ and $\Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*)$ directly, without first computing the features, we could do computations without incurring the cost for large features!

Example: Polynomial kernel

$$k(x, y) = (xy + 1)^{M-1} = \sum_{m=0}^{M-1} \binom{M-1}{m} x^m y^m = \boldsymbol{\phi}(x)^\intercal \boldsymbol{\phi}(y) \tag{24}$$

$$\text{for } M = 3, \quad \boldsymbol{\phi}(x) = \begin{bmatrix} 1 & \sqrt{2}x & x^2 \end{bmatrix}^\intercal \tag{25}$$

Kernels: Polynomial kernel

If we can compute the matrices $\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal$ and $\Phi(\boldsymbol{X})\boldsymbol{\phi}(\mathbf{x}_*)$ directly, without first computing the features, we could do computations without incurring the cost for large features!

Example: Polynomial kernel

$$k(x, y) = (xy + 1)^{M-1} = \sum_{m=0}^{M-1} \binom{M-1}{m} x^m y^m = \boldsymbol{\phi}(x)^\intercal \boldsymbol{\phi}(y) \tag{24}$$

$$\text{for } M = 3, \quad \boldsymbol{\phi}(x) = \begin{bmatrix} 1 & \sqrt{2}x & x^2 \end{bmatrix}^\intercal \tag{25}$$

We can compute very large inner products for very cheap!

Kernels: Infinite Dimensional Feature Spaces

We can even consider infinite dimensional feature spaces, if the limit of the inner product exists!

$$\phi_m(x) = \exp\left(-\frac{(x-c_m)^2}{2\ell^2}\right), \qquad c_m = \frac{m}{M}\cdot(c_{\max}-c_{\min}) \tag{26}$$

$$k(x,x') = \frac{1}{M}\sum_{m=1}^{M}\phi_m(x)\phi_m(x')$$

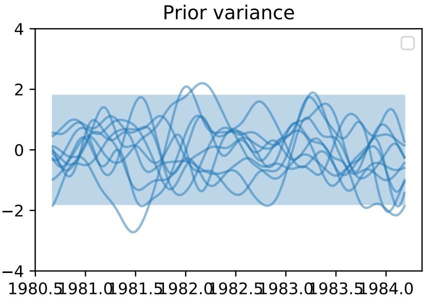
$$\begin{aligned}\lim_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M}\phi_m(x)\phi_m(x') &= \int_{c_{\min}}^{c_{\max}}\exp\left(-\frac{(x-c)^2}{2\ell^2}\right)\exp\left(-\frac{(x'-c)^2}{2\ell^2}\right)\mathrm{d}c \\ &= \sqrt{\pi}\ell\exp\left(-\frac{(x-x')^2}{4\ell^2}\right)\end{aligned}$$

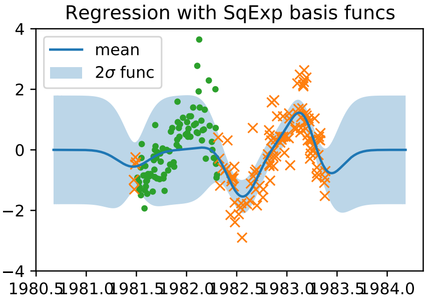
Squared Exponential Kernel: Infinite SqExp basis functions, everywhere!

Gaussian Process Prediction

So how do we do prediction? Just replace inner products $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ with $k(\mathbf{x}, \mathbf{x}')$. Now cost is $O(N^3 + N^2) = O(N^3)$, down from ∞ for basis funcs.

$$p(y_*|\mathbf{y}) = \mathcal{N}\Big(y_*; \quad k(\mathbf{x}_*, \boldsymbol{X})\left[k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_N\right]^{-1}\mathbf{y}\,, \\ k(\mathbf{x}_*, \mathbf{x}_*) + \sigma^2 - k(\mathbf{x}_*, \boldsymbol{X})\left[k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}_N\right]^{-1} k(\boldsymbol{X}, \mathbf{x}_*)\Big) \qquad (27)$$





Recap

What did we do?

1. Start with a basis function model.
2. **Integrated out parameters** to directly find **predictive distribution** $p(y_* | \mathbf{y})$.
3. Prediction only depended on **inner products** of feature vectors.
4. We showed that we could compute inner products with a **kernel function**.
5. Computational cost down from ∞ to $O(N^3)$.
6. Different **representation** of a basis function model.

Recap

What did we do?

1. Start with a basis function model.
2. **Integrated out parameters** to directly find **predictive distribution** $p(y_*|\mathbf{y})$.
3. Prediction only depended on **inner products** of feature vectors.
4. We showed that we could compute inner products with a **kernel function**.
5. Computational cost down from ∞ to $O(N^3)$.
6. Different **representation** of a basis function model.

... but what is a Gaussian process?

Priors on Function Values

Another way of looking at our model:

$$p(y_n|f(\mathbf{x}_n), \mathbf{x}_n) = \mathcal{N}\left(y_n; f(\mathbf{x}_n), \sigma^2\right) \tag{28}$$
$$p(f(\boldsymbol{X})) = \mathcal{N}(f(\boldsymbol{X}); \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{29}$$

Remember: Each parameter *implied* an entire function. So our prior placed a distribution on all the function values.

For a basis function model, find the prior on the vector of function values at each input point, denoted $f(\boldsymbol{X})$, from the prior on the weights $p(\boldsymbol{\theta}) = \mathcal{N}(0, \boldsymbol{I}_M)$

$$\boldsymbol{\Sigma} = \mathbb{V}_{p(\boldsymbol{\theta})}[\Phi(\boldsymbol{X})\boldsymbol{\theta}] = \Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\intercal \tag{30}$$

Priors on Function Values

Another way of looking at our model:

$$p(y_n|f(\mathbf{x}_n), \mathbf{x}_n) = \mathcal{N}\left(y_n; f(\mathbf{x}_n), \sigma^2\right) \tag{28}$$

$$p(f(\boldsymbol{X})) = \mathcal{N}(f(\boldsymbol{X}); \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{29}$$

Remember: Each parameter *implied* an entire function. So our prior placed a distribution on all the function values.

For a basis function model, find the prior on the vector of function values at each input point, denoted $f(\boldsymbol{X})$, from the prior on the weights $p(\boldsymbol{\theta}) = \mathcal{N}(0, \boldsymbol{I}_M)$

$$\boldsymbol{\Sigma} = \mathbb{V}_{p(\boldsymbol{\theta})}[\Phi(\boldsymbol{X})\boldsymbol{\theta}] = \Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top \tag{30}$$

A Gaussian process specifies $[\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^\top]_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$ **directly**:

$$p(f(X)) = \mathcal{N}(f(X); 0, k(\boldsymbol{X}, \boldsymbol{X})) \tag{31}$$

So what really is a Gaussian process?

See handwritten notes for:

- Definition of Gaussian process
- Gaussian processes as distributions on functions
- BLR defines a Gaussian process
- Find the posterior of a GP

Recommended reading

- Rasmussen and Williams (2006) §2.1 + §2.2

References I

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*. MIT press, Cambridge, MA, USA.