Kernel Machines

Andrea Passerini
passerini@disi.unitn.it

Machine Learning

Non-linear Support Vector Machines

Non-linearly separable problems

- Hard-margin SVM can address linearly separable problems
- Soft-margin SVM can address linearly separable problems with outliers
- Non-linearly separable problems need a higher expressive power (i.e. more complex feature combinations)
- We do not want to lose the advantages of linear separators (i.e. large margin, theoretical guarantees)

Solution

- Map input examples in a higher dimensional *feature space*
- Perform linear classification in this higher dimensional space

Non-linear Support Vector Machines

feature map

$$\Phi : \mathcal{X} \rightarrow \mathcal{H}$$

- Φ is a function mapping each example to a higher dimensional space $\mathcal{H}$
- Examples $\boldsymbol{x}$ are replaced with their feature mapping $\Phi(\boldsymbol{x})$
- The feature mapping should increase the expressive power of the representation (e.g. introducing features which are combinations of input features)
- Examples should be (approximately) linearly separable in the mapped space

Feature map

Homogeneous ($d = 2$)

$$\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}$$

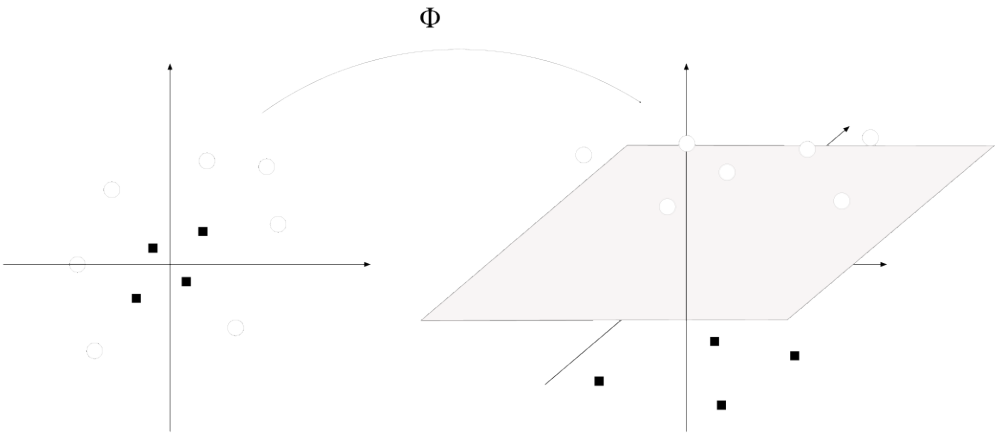
Inhomogeneous ($d = 2$)

$$\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}$$

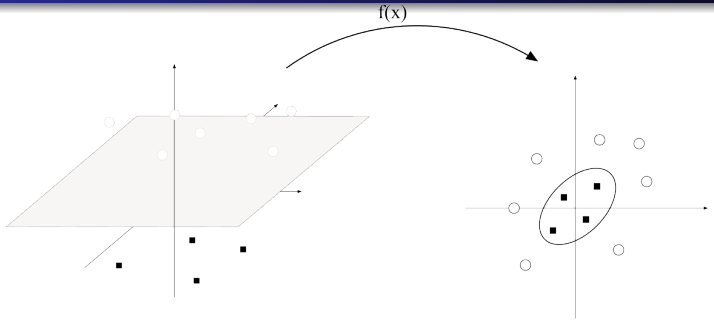
Polynomial mapping

- Maps features to all possible conjunctions (i.e. products) of features:
  1. *of a certain degree d* (*homogeneous* mapping)
  2. *up to a certain degree* (*inhomogeneous* mapping)

Feature map



Non-linear Support Vector Machines



Linear separation in feature space

- SVM algorithm is applied just replacing $\boldsymbol{x}$ with $\Phi(\mathbf{x})$:

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + w_0$$

- A linear separation (i.e. hyperplane) in feature space corresponds to a non-linear separation in input space, e.g.:

Kernel trick

- Feature mapping $\Phi(\cdot)$ can be very high dimensional (e.g. think of polynomial mapping)
- It can be highly expensive to explicitly compute it
- Feature mappings **appear only in dot products** in dual formulations
- The *kernel trick* consists in replacing these dot products with an equivalent kernel function:

$$k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$$

- The kernel function uses examples in input (not feature) space

Support vector classification

- Dual optimization problem

$$\begin{aligned} \max_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \underbrace{\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)}_{k(\mathbf{x}_i, \mathbf{x}_j)} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

- Dual decision function

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i \underbrace{\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x})}_{k(\mathbf{x}_i, \mathbf{x})}$$

Polynomial kernel

- Homogeneous:

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^d$$

- E.g. ($d = 2$)

$$\begin{aligned} k\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}\right) &= (x_1 x_1' + x_2 x_2')^2 \\ &= (x_1 x_1')^2 + (x_2 x_2')^2 + 2 x_1 x_1' x_2 x_2' \\ &= \underbrace{\begin{pmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{pmatrix}}_{\Phi(\mathbf{x})^T} \underbrace{\begin{pmatrix} x_1'^2 \\ \sqrt{2} x_1' x_2' \\ x_2'^2 \end{pmatrix}}_{\Phi(\mathbf{x}')} \end{aligned}$$

Kernel trick

Polynomial kernel

- Inhomogeneous: $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^d$
- E.g. ($d = 2$)

$$
\begin{aligned}
k(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}) &= (1 + x_1 x_1' + x_2 x_2')^2 \\
&= 1 + (x_1 x_1')^2 + (x_2 x_2')^2 + 2 x_1 x_1' + 2 x_2 x_2' + 2 x_1 x_1' x_2 x_2' \\
&= \underbrace{\begin{pmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & x_1^2 & \sqrt{2}x_1 x_2 & x_2^2 \end{pmatrix}}_{\Phi(\mathbf{x})^T} \underbrace{\begin{pmatrix} 1 \\ \sqrt{2}x_1' \\ \sqrt{2}x_2' \\ x_1'^2 \\ \sqrt{2}x_1' x_2' \\ x_2'^2 \end{pmatrix}}_{\Phi(\mathbf{x}')}
\end{aligned}
$$

Valid Kernels

Dot product in feature space

- A valid kernel is a (similarity) function defined in cartesian product of input space:

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$$

- corresponding to a dot product in a (certain) feature space:

$$k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$$

Note

- The kernel generalizes the notion of dot product to arbitrary input space (e.g. protein sequences)
- It can be seen as a measure of similarity between objects

Valid Kernels

Gram matrix

- Given examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and kernel function k
- The *Gram matrix* K is the (symmetric) matrix of pairwise kernels between examples:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \quad \forall i, j$$

Valid Kernels

Positive definite matrix

- A symmetric $m \times m$ matrix K is *positive definite* (p.d.) if

$$\sum_{i,j=1}^{m} c_i c_j K_{ij} \geq 0, \quad \forall \mathbf{c} \in \mathbb{R}^m$$

If equality only holds for $\boldsymbol{c} = \mathbf{0}$, the matrix is *strictly positive definite* (s.p.d)

Alternative conditions

- All eigenvalues are non-negative (positive for s.p.d.)
- There exists a matrix B such that

$$K = B^T B$$

Valid Kernels

Positive definite kernels

- A positive definite kernel is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ giving rise to a p.d. Gram matrix for any m and $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$
- Positive definiteness is necessary and sufficient condition for a kernel to correspond to a dot product of *some* feature map Φ

How to verify kernel validity

- Prove its positive definiteness (difficult)
- Find out a corresponding feature map (see polynomial example)
- Use kernel combination properties (we'll see)

Kernels

Basic kernels

- linear kernel:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- polynomial kernel:

$$k_{d,c}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d$$

Gaussian kernel

$$k_\sigma(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) = \exp\left(-\frac{\mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{x}' + \mathbf{x}'^T\mathbf{x}'}{2\sigma^2}\right)$$

- Depends on a *width* parameter σ
- The smaller the width, the more prediction on a point only depends on its nearest neighbours
- Example of *Universal* kernel: they can uniformly approximate any arbitrary continuous target function (pb of number of training examples and choice of σ)

From Kernels to Gaussian Processes

A kernel can define more than similarity

A positive definite kernel can also be interpreted as a *covariance function* between function values:

$$k(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}\big[f(\mathbf{x}), f(\mathbf{x}')\big] .$$

- Similar inputs have strongly correlated function values.
- The kernel encodes assumptions such as smoothness, periodicity, or linearity.
- This leads to a Bayesian model over entire functions.

Key connection

SVMs use kernels inside an optimization problem; Gaussian processes use kernels to define a probability distribution over functions.

Gaussian Processes

Distribution over functions

A Gaussian process is specified by a mean function and a kernel:

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big) .$$

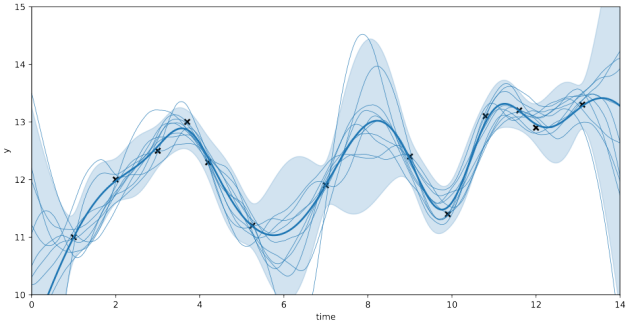
For any finite collection $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$,

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_m) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_m) \end{bmatrix}, K \right), \qquad K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j).$$

Interpretation

A sample from a Gaussian process is a function. Different kernels produce different families of plausible functions.

Gaussian Process Regression



Posterior distribution over functions

Conditioning GP prior on data $\rightarrow$ **posterior distribution over functions**.

- The posterior mean is used as the prediction.
- The shaded confidence region represents the posterior uncertainty.
- Uncertainty is smallest near training points and grows away from them.

Gaussian Process Prediction

Let

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \qquad \mathbf{k}_* = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_*) & \cdots & k(\mathbf{x}_m, \mathbf{x}_*) \end{bmatrix}^T.$$

For a test input $\mathbf{x}_*$, the posterior predictive distribution is

$$f_* \mid \mathbf{x}_*, X, \boldsymbol{y} \sim \mathcal{N}\left(\mu_*, \sigma_*^2\right),$$

with

$$\boxed{\mu_* = \mathbf{k}_*^T (K + \sigma_n^2 I)^{-1} \boldsymbol{y}}$$

and

$$\boxed{\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \sigma_n^2 I)^{-1} \mathbf{k}_*}.$$

Interpretation

- $\mathbf{k}_*^T (K + \sigma_n^2 I)^{-1} \boldsymbol{y}$ is a kernel-weighted combination of observed outputs.
- $k(\mathbf{x}_*, \mathbf{x}_*)$ is the prior uncertainty at $\boldsymbol{x}_*$.
- The subtracted term is the uncertainty removed by observing the training data.

Gaussian Processes: Role of the Kernel

Kernel choice expresses prior assumptions

- **Linear kernel:** approximately linear functions.
- **Gaussian/RBF kernel:** smooth functions; the width controls how rapidly the function may vary.
- **Periodic kernel:** repeating patterns.
- **Kernel sums and products:** combine several structural assumptions.

Learning the kernel parameters

Hyperparameters such as the RBF width and noise variance are often selected by maximizing the marginal likelihood $p(\boldsymbol{y} \mid X)$.

Computational limitation

Exact GP regression requires solving a linear system involving the $m \times m$ Gram matrix: typically $\mathcal{O}(m^3)$ time and $\mathcal{O}(m^2)$ memory.

Kernels

Kernels on structured data

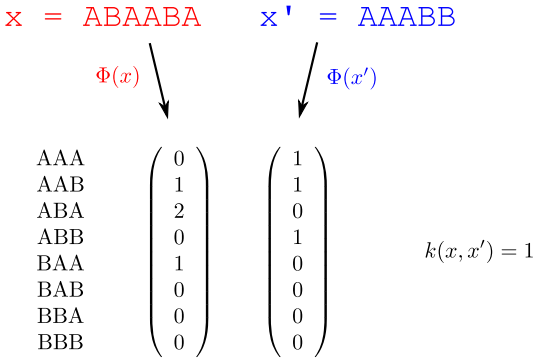
- Kernels are generalization of dot products to arbitrary domains
- It is possible to design kernels over structured objects like sequences, trees or graphs
- The idea is designing a pairwise function measuring the similarity of two objects
- This measure has to satisfy the p.d. conditions to be a valid kernel

Match (or delta) kernel

$$k_\delta(x, x') = \delta(x, x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise.} \end{cases}$$

- Simplest kernel on structures
- x does not need to be a vector! (no boldface to stress it)

Kernels on sequences



Spectrum kernel

- Feature space is space of all possible k-grams (subsequences)
- An efficient procedure based on suffix trees allows to compute kernel without explicitly building feature maps

Kernels on graphs

Weisfeiler–Lehman graph kernel

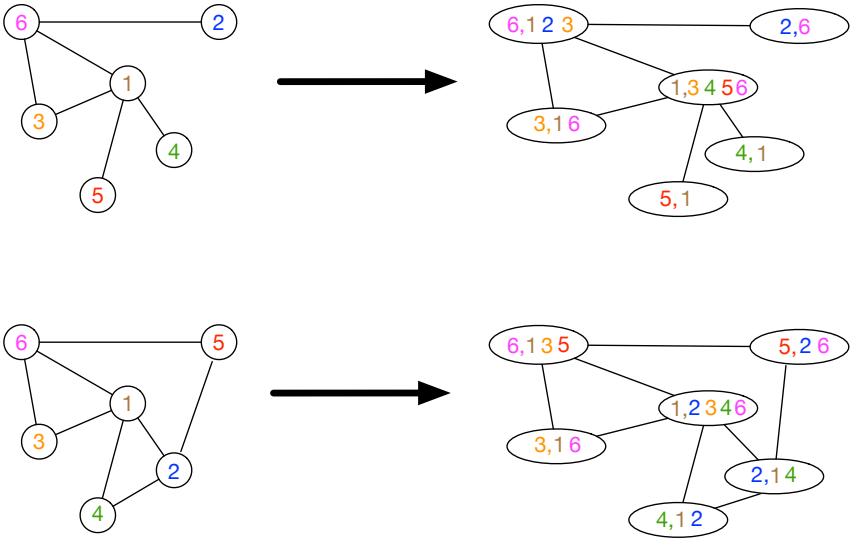
- Efficient graph kernel for large graphs
- Relies on (approximation of) Weisfeiler–Lehman test of graph isomorphism
- Defines a family of graph kernels

Weisfeiler–Lehman (WL) isomorphism test

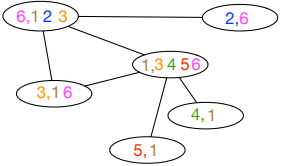
Given $G = (\mathcal{V}, \mathcal{E})$ and $G' = (\mathcal{V}', \mathcal{E}')$, with $n = |\mathcal{V}| = |\mathcal{V}'|$. Let $L(G) = \{l(v) | v \in \mathcal{V}\}$ be the set of labels in G, and let $L(G) == L(G')$. Let $label(s)$ be a function assigning a unique label to a string.

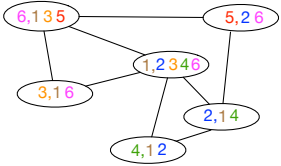
- Set $l_0(v) = l(v)$ for all v.
- For $i \in [1, n-1]$
  1. For each node v in G and G'
  2. Let $M_i(v) = \{l_{i-1}(u) | u \in neigh(v)\}$
  3. Concatenate the sorted labels of $M_i(v)$ into $s_i(v)$
  4. Let $l_i(v) = label(l_{i-1}(v) \circ s_i(v))$ ($\circ$ is concatenation)
  5. If $L_i(G) \neq L_i(G')$
  6. Return **Fail**
- Return **Pass**

WL isomorphism test: string determination



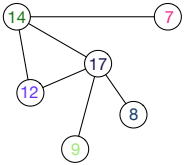
WL isomorphism test: relabeling

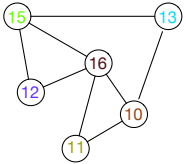




2,6 → 7
4,1 → 8
5,1 → 9
2,1 4 → 10
4,1 2 → 11
3,1 6 → 12

5,2 6 → 13
6,1 2 3 → 14
6,1 3 5 → 15
1,2 3 4 6 → 16
1,3 4 5 6 → 17



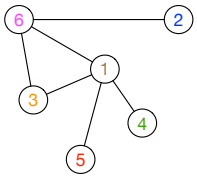


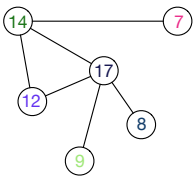
Weisfeiler–Lehman graph kernel

- Let $\{G_0, G_1, \ldots, G_h\} = \{(\mathcal{V}, \mathcal{E}, l_0), (\mathcal{V}, \mathcal{E}, l_1), \ldots, (\mathcal{V}, \mathcal{E}, l_h)\}$ be a sequence of graphs made from G, where l_i is the node labeling of the i-th WL iteration.
- Let $k : G \times G' \to \mathbb{R}$ be any kernel on graphs.
- The Weisfeiler–Lehman graph kernel is defined as:

$$k_{WL}^{h}(G, G') = \sum_{i=0}^{h} k(G_i, G_i')$$

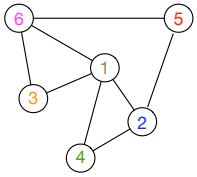
Example: WL subtree kernel

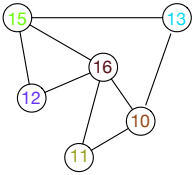




1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

(1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1)





(1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0)

References

kernel trick C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery, 2(2), 121-167, 1998.

kernel properties J.Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004 (Section 3)

kernels J.Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004 (Section 9)

gaussian processes C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

graph kernels N. Shervashidze, P. Schweitzer, E. Jan van Leeuwen, K. Mehlhorn, and K. Borgwardt. *Weisfeiler-Lehman Graph Kernels*. J. Mach. Learn. Res., 2011.

Appendix

Additional reference material

GP Regression: Setup

Assume a Gaussian-process prior

$$f(\boldsymbol{x}) \sim \mathcal{GP}\big(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\big)\,.$$

For clarity, take $m(\boldsymbol{x}) = 0$. At the training inputs

$$X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix}, \qquad \boldsymbol{f} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_n) \end{bmatrix},$$

the GP prior implies

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, K), \qquad K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

GP Regression: Setup

Observations are corrupted by independent Gaussian noise:

$$\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 I).$$

Therefore,

$$\boxed{\boldsymbol{y} \sim \mathcal{N}\left(\mathbf{0}, K + \sigma_n^2 I\right)}.$$

GP Regression: Joint Gaussian Distribution

Consider a new input $\boldsymbol{x}_*$, and define $f_* = f(\boldsymbol{x}_*)$

$$\mathbf{k}_* = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_*) \\ \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_*) \end{bmatrix}, \qquad k_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*).$$

Since every finite collection of GP values is jointly Gaussian,

$$\begin{bmatrix} \boldsymbol{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} K + \sigma_n^2 I & \mathbf{k}_* \\ \mathbf{k}_*^T & k_{**} \end{bmatrix} \right).$$

Why is the cross-covariance $\mathbf{k}_*$?

Observation noise is independent of the latent function, so

$$\mathrm{Cov}(\boldsymbol{y}, f_*) = \mathrm{Cov}(\boldsymbol{f} + \epsilon, f_*) = \mathrm{Cov}(\boldsymbol{f}, f_*) = \mathbf{k}_*.$$

Conditioning a Joint Gaussian

Suppose

$$\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \right).$$

Then the conditional distribution of $\boldsymbol{b}$ given $\boldsymbol{a}$ is

$$\boldsymbol{b} \mid \boldsymbol{a} \sim \mathcal{N}\left(\boldsymbol{\mu}_b + C^T A^{-1}(\boldsymbol{a} - \boldsymbol{\mu}_a),\ B - C^T A^{-1} C \right).$$

For GP regression:

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{y}, & \boldsymbol{b} &= f_*, & \boldsymbol{\mu}_a &= \mathbf{0}, & \boldsymbol{\mu}_b &= 0, \\ A &= K + \sigma_n^2 I, & C &= \mathbf{k}_*, & B &= k_{**}. \end{aligned}$$

Conditioning on the observed values $\boldsymbol{y} = \boldsymbol{y}_{\text{obs}}$ therefore gives the posterior predictive distribution.

GP Posterior Mean and Variance

Applying the conditional-Gaussian identity gives

$$f_* \mid X, \boldsymbol{y}, \boldsymbol{x}_* \sim \mathcal{N}\left(\mu_*, \sigma_*^2\right),$$

where the posterior mean is

$$\boxed{\mu_* = \mathbf{k}_*^T \left(K + \sigma_n^2 I\right)^{-1} \boldsymbol{y}}$$

and the posterior variance is

$$\boxed{\sigma_*^2 = k_{**} - \mathbf{k}_*^T \left(K + \sigma_n^2 I\right)^{-1} \mathbf{k}_*}.$$