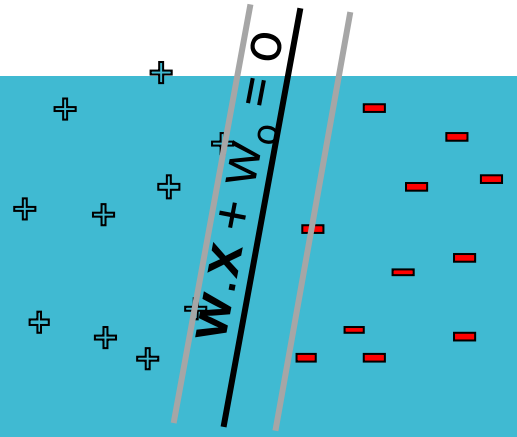
10-701: Introduction to Machine Learning Lecture 25 – Kernels & Gaussian Processes

Pradeep Ravikumar & Geoff Gordon

Spring 2026

Front Matter

- Announcements:
  - HW6
    - due on Wednesday, April 22
    - late due date on Friday, April 24
  - Project
    - Video Showcase due on Wednesday, April 22
    - Final report due on Thursday, April 23
    - Project video showcase this Friday, April 24! Attendance required, refreshments will be served!
    - As a reminder, no late days can be used for project deliverables.
- Recommended Readings:
  - Murphy, Chapters 14.1-14.5



Recap: SVMs, Primal-Dual Optimization

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\text{subject to} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + w_0\right) \geq 1 \; \forall \left(\boldsymbol{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}$$

Primal

$$\Updownarrow$$

$$\text{maximize} \quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\boldsymbol{x}^{(i)^T}\boldsymbol{x}^{(j)} + \sum_{i=1}^{N}\alpha^{(i)}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

$$\alpha^{(i)} \geq 0 \; \forall \; i \in \{1, \ldots, N\}$$

Dual

Recap: SVMs, Primal-Dual Optimization

- Primal
  - Directly returns the weights, $[\hat{w}_0, \hat{\boldsymbol{w}}]$
  - Support vectors are all $\left(\boldsymbol{x}^{(s)}, y^{(s)}\right) \in \mathcal{D}$ s.t.

$$y^{(s)}\left(\hat{\boldsymbol{w}}^T \boldsymbol{x}^{(s)} + \hat{w}_0\right) = 1$$

- Dual
  - Returns the vector, $\hat{\boldsymbol{\alpha}}$

$$\hat{\boldsymbol{w}} = \sum_{i=1}^{N} \hat{\alpha}^{(i)} y^{(i)} \boldsymbol{x}^{(i)}$$

$$\hat{w}_0 = y^{(s)} - \hat{\boldsymbol{w}}^T \boldsymbol{x}^{(s)} \text{ for any } s \text{ s.t. } \hat{\alpha}^{(s)} > 0$$

  - Support vectors are all $\left(\boldsymbol{x}^{(s)}, y^{(s)}\right) \in \mathcal{D}$ s.t. $\hat{\alpha}^{(s)} > 0$

Primal-Dual Optimization

- Primal
  - $\hat{y} = \text{sign}(\hat{\boldsymbol{w}}^T \vec{x} + \hat{w}_0)$

- Dual
  - $\hat{y} = \text{sign}(\hat{\boldsymbol{w}}^T \vec{x} + \hat{w}_0)$

$$= \text{sign}\left(\left(\sum_{i=1}^{N} \hat{\alpha}^{(i)} y^{(i)} \boldsymbol{x}^{(i)}\right)^T \boldsymbol{x} + \hat{w}_0\right)$$

$$= \text{sign}\left(\sum_{i\,:\,\hat{\alpha}^{(i)} > 0} \hat{\alpha}^{(i)} y^{(i)} \boldsymbol{x}^{(i)^T} \boldsymbol{x} + \hat{w}_0\right)$$

Primal-Dual Soft-Margin SVMs

Primal

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi^{(i)}$$

$$\text{subject to} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + w_0\right) \geq 1 - \xi^{(i)} \quad \forall \left(\boldsymbol{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}$$

$$\xi^{(i)} \geq 0 \quad \forall\, i \in \{1, \ldots, N\}$$

Dual

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\boldsymbol{x}^{(i)^T}\boldsymbol{x}^{(j)} - \sum_{i=1}^{N}\alpha^{(i)}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

$$0 \leq \alpha^{(i)} \leq C \;\forall\, i \in \{1, \ldots, N\}$$

Primal-Dual Soft-Margin SVMs

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi^{(i)}$$

$$\text{subject to} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + w_0\right) \geq 1 - \xi^{(i)} \quad \forall \left(\boldsymbol{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}$$

$$\xi^{(i)} \geq 0 \quad \forall\, i \in \{1, \ldots, N\}$$

Primal

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\boldsymbol{x}^{(i)^T}\boldsymbol{x}^{(j)} - \sum_{i=1}^{N}\alpha^{(i)}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

$$0 \leq \alpha^{(i)} \leq C \;\forall\, i \in \{1, \ldots, N\}$$

Dual

Recall: Nonlinear Transforms

- Decide on some transformation $\Phi: \mathcal{X} \to \mathcal{Z}$
- Given $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$, learn a hypothesis, $\tilde{h}(\boldsymbol{z})$, using $\widetilde{\mathcal{D}} = \left\{\left(\boldsymbol{z}^{(i)} = \Phi\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)\right\}_{i=1}^{N}$
- Return the corresponding predictor in the original space: $h(\boldsymbol{x}) = \tilde{h}\left(\Phi(\boldsymbol{x})\right)$

Nonlinear SVMs

- Decide on some transformation $\Phi: \mathcal{X} \rightarrow \mathcal{Z}$
- Find a maximal-margin separating hyperplane in the transformed space, $[\widetilde{\widehat{\boldsymbol{w}}}, \widetilde{\widehat{w}_0}]$, by solving the QP:

$$\text{minimize} \quad \frac{1}{2}\widetilde{\boldsymbol{w}}^T\widetilde{\boldsymbol{w}}$$

$$\text{subject to} \quad y^{(i)}\left(\widetilde{\boldsymbol{w}}^T\Phi\left(\boldsymbol{x}^{(i)}\right) + \widetilde{w_0}\right) \geq 1 \; \forall \left(\boldsymbol{x}^{(i)}, y^{(i)}\right) \in \mathcal{D}$$

- Return the corresponding predictor in the original space:

$$h(\boldsymbol{x}) = \text{sign}\left(\widetilde{\widehat{\boldsymbol{w}}}^T\Phi(\boldsymbol{x}) + \widetilde{\widehat{w}_0}\right)$$

Nonlinear Dual SVMs

- Decide on some transformation $\Phi: \mathcal{X} \to \mathcal{Z}$
- Find a maximal-margin separating hyperplane in the transformed space by solving the QP:

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\Phi\left(\boldsymbol{x}^{(i)}\right)^{T}\Phi\left(\boldsymbol{x}^{(j)}\right) - \sum_{i=1}^{N}\alpha^{(i)}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

$$0 \leq \alpha^{(i)} \leq C \;\forall\; i \in \{1, \ldots, N\}$$

- Return the corresponding predictor in the original space:

$$h(\boldsymbol{x}) = \text{sign}\left(\sum_{i:\hat{\alpha}^{(i)} > 0}\hat{\alpha}^{(i)}y^{(i)}\Phi\left(\boldsymbol{x}^{(i)}\right)^{T}\Phi(\boldsymbol{x}) + \widetilde{\hat{w}_0}\right)$$

Efficiency

- Depending on the transformation Φ and the dimensionality of the original input space $\mathcal{X}$, computing $\Phi(\boldsymbol{x})$ can be prohibitively computationally expensive
  - Computing $\Phi_2(\boldsymbol{x}) = [x_1, x_2, \dots, x_D, x_1^2, x_1 x_2, \dots, x_D^2]$ requires $D + \binom{D}{2} + D = \frac{D^2+3D}{2} = O(D^2)$ time
  - Computing $\Phi_{10}(\boldsymbol{x})$ requires $O(D^{10})$ time
- Tradeoff:
  - High-dimensional transformations can result in good hypotheses (as long as they don't overfit)
  - High-dimensional transformations are expensive

Nonlinear Dual SVMs

- Insight: Φ only appears in inner products!
- Find a maximal-margin separating hyperplane in the transformed space by solving the QP:

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\Phi\left(\boldsymbol{x}^{(i)}\right)^{T}\Phi\left(\boldsymbol{x}^{(j)}\right) - \sum_{i=1}^{N}\alpha^{(i)}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

$$0 \leq \alpha^{(i)} \leq C \;\forall\; i \in \{1, \ldots, N\}$$

- Return the corresponding predictor in the original space:

$$h(\boldsymbol{x}) = \text{sign}\left(\sum_{i:\hat{\alpha}^{(i)}>0}\hat{\alpha}^{(i)}y^{(i)}\Phi\left(\boldsymbol{x}^{(i)}\right)^{T}\Phi(\boldsymbol{x}) + \widetilde{\hat{w}_0}\right)$$

Nonlinear Dual SVMs

- Insight: Φ only appears in inner products!
- Find a maximal-margin separating hyperplane in the transformed space by solving the QP:

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\Phi\left(\boldsymbol{x}^{(i)}\right)^{T}\Phi\left(\boldsymbol{x}^{(j)}\right) - \sum_{i=1}^{N}\alpha^{(i)}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

$$0 \leq \alpha^{(i)} \leq C \;\forall\; i \in \{1, \ldots, N\}$$

- Return the corresponding predictor in the original space:

$$h(\boldsymbol{x}) = \text{sign}\left(\sum_{i:\hat{\alpha}^{(i)}>0}\hat{\alpha}^{(i)}y^{(i)}\Phi\left(\boldsymbol{x}^{(i)}\right)^{T}\Phi(\boldsymbol{x}) + \widetilde{\widehat{w}_0}\right)$$

The Kernel Trick

- Approach: instead of computing $\Phi(\boldsymbol{x})$, find some function K_Φ s.t. $K_\Phi(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}') \; \forall \, \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$
  - $K_\Phi(\boldsymbol{x}, \boldsymbol{x}')$ should be cheaper to compute than $\Phi(\boldsymbol{x})$
- Example: $\Phi_2'(\boldsymbol{x}) = \left[x_1, \ldots, x_D, x_1^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{D-1}x_D, x_D^2\right]$

$$\Phi_2'(\boldsymbol{x})^T \Phi_2'(\boldsymbol{x}') = \sum_{i=1}^{D} x_i x_i' + \sum_{i=1}^{D} x_i^2 {x_i'}^2 + \sum_{i=1}^{D} \sum_{j>i} 2 x_i x_i' x_j x_j'$$

$$= \sum_{i=1}^{D} x_i x_i' + \left(\sum_{i=1}^{D} x_i x_i' \right)^2 = \boldsymbol{x}^T \boldsymbol{x}' + (\boldsymbol{x}^T \boldsymbol{x}')^2$$

$$K_{\Phi_2'}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}' + (\boldsymbol{x}^T \boldsymbol{x}')^2$$

- Computing $\Phi_2'(\boldsymbol{x})^T \Phi_2'(\boldsymbol{x}')$ requires $O(D^2)$ time whereas computing $K_{\Phi_2'}(\boldsymbol{x}, \boldsymbol{x}')$ only takes $O(D)$!

Common Kernels

- $K_{\Phi_2'}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T\boldsymbol{x}' + (\boldsymbol{x}^T\boldsymbol{x}')^2$
  - Implied feature transformation:

$$\Phi_2'(\boldsymbol{x}) = \left[x_1, \ldots, x_D, x_1^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{D-1}x_D, x_D^2\right]$$

  - Implied dimensionality: $\frac{D^2+3D}{2}$

- $K_{\Phi_2^{(\gamma)}}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \gamma\boldsymbol{x}^T\boldsymbol{x}')^2 - 1$
  - Implied feature transformation:

$$\Phi_2^{(\gamma)}(\boldsymbol{x}) = \left[\sqrt{2\gamma}x_1, \ldots, \sqrt{2\gamma}x_D, \gamma x_1^2, \gamma x_1x_2, \ldots, \gamma x_D^2\right]$$

    - γ affects the geometry of the transform

  - Implied dimensionality: $\frac{D^2+3D}{2}$

Common Kernels

- Polynomial Kernel: $K_{\Phi_Q^{(\gamma)}}(\boldsymbol{x}, \boldsymbol{x}') = (1 + \gamma \boldsymbol{x}^T \boldsymbol{x}')^Q - 1$
  - Implied dimensionality: $O(D^Q)$
  - γ affects the geometry of the transform
- Gaussian-RBF Kernel: $K_{\Phi_r}(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|x-x'\|^2}{2r}}$
  - Implied feature transformation: $\Phi_r(x) =$

$$\left[\left[e^{-\frac{x_1^2}{2r}}\sqrt{\frac{\left(x_1^d\right)^2}{d!r^d}}, \ldots, e^{-\frac{x_D^2}{2r}}\sqrt{\frac{\left(x_1^d\right)^2}{d!r^d}}\right] : d \in \mathbb{N}\right]$$

  - Implied dimensionality: ∞!

Nonlinear Dual SVMs

- Decide on a (valid) kernel function K_Φ
- Find a maximal-margin separating hyperplane in the transformed space by solving the QP:

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}K_\Phi\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}\right)-\sum_{i=1}^{N}\alpha^{(i)}$$

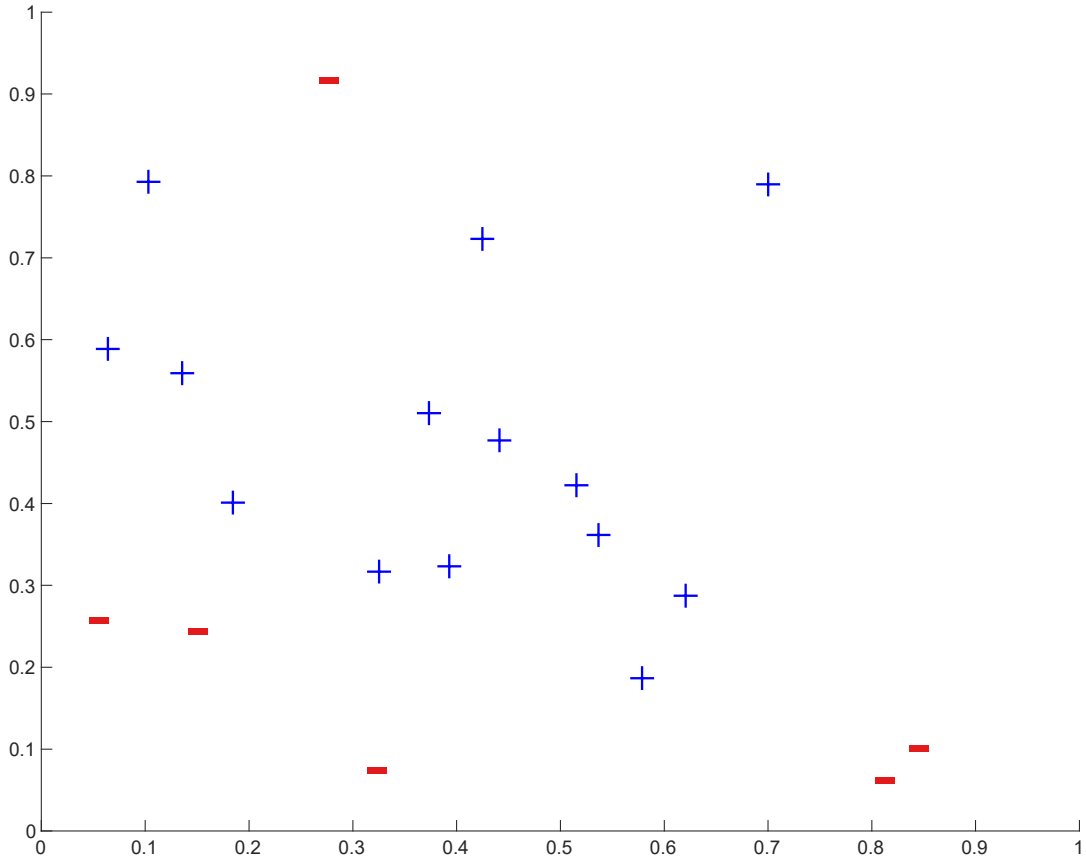
$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}=0$$

$$0 \le \alpha^{(i)} \;\forall\, i \in \{1,\dots,N\}$$

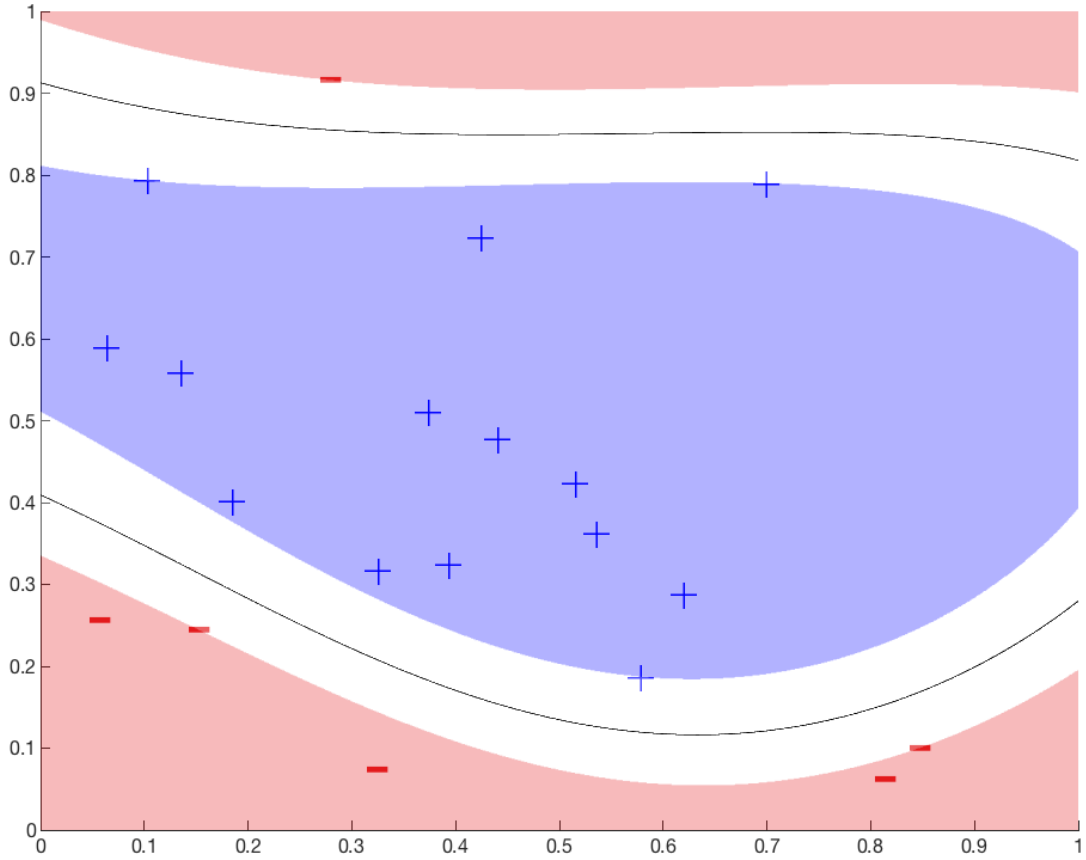
- Return the corresponding predictor in the original space:

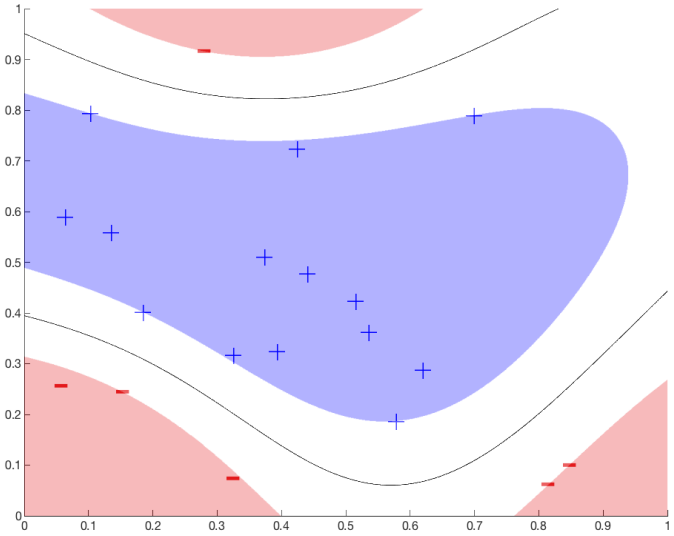
$$h(\boldsymbol{x}) = \text{sign}\left(\sum_{i:\hat{\alpha}^{(i)}>0}\hat{\alpha}^{(i)}y^{(i)}K_\Phi\left(\boldsymbol{x}^{(i)},\boldsymbol{x}\right)+\widetilde{\hat{w}_0}\right)$$

Gaussian-RBF Kernel

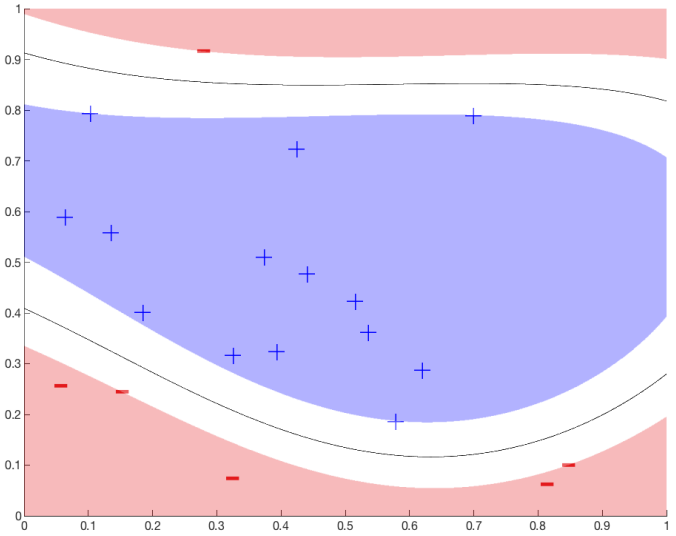


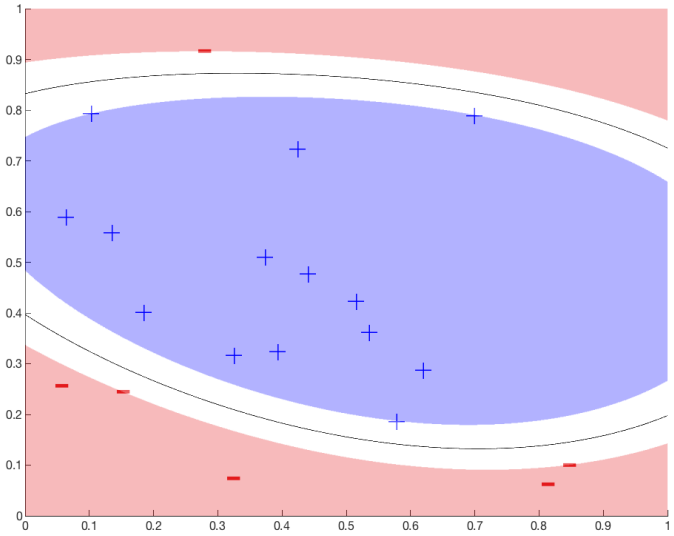
Gaussian-RBF Kernel





Smaller r





Larger r

Gaussian-RBF Kernel

Nonlinear Dual Soft-Margin SVMs

- Decide on a (valid) kernel function K_Φ
- Find a maximal-margin separating hyperplane in the transformed space by solving the QP:

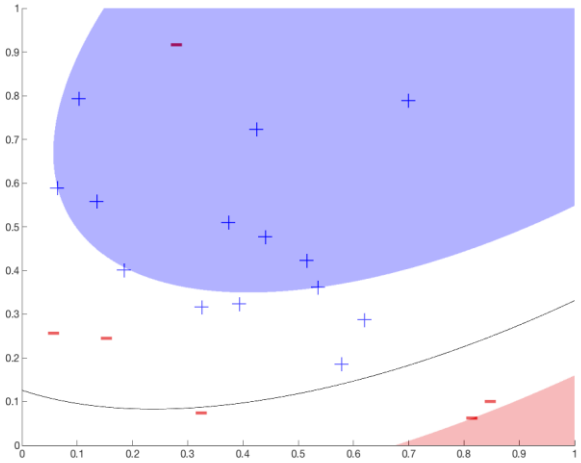
$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}K_\Phi\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}\right)-\sum_{i=1}^{N}\alpha^{(i)}$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}=0$$

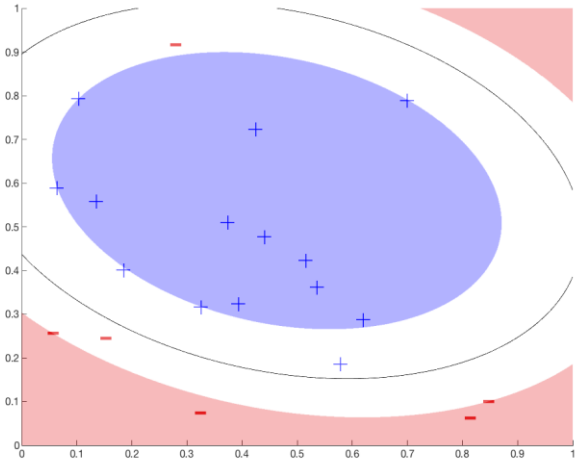
$$0\le\alpha^{(i)}\le C\ \forall\ i\in\{1,\dots,N\}$$

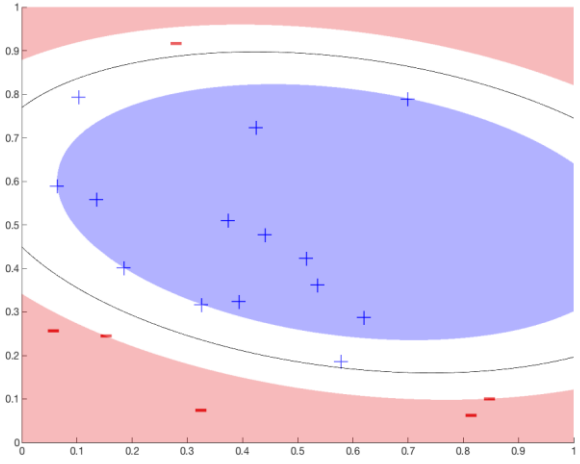
- Return the corresponding predictor in the original space:

$$h(\boldsymbol{x})=\text{sign}\left(\sum_{i:\hat{\alpha}^{(i)}>0}\hat{\alpha}^{(i)}y^{(i)}K_\Phi\left(\boldsymbol{x}^{(i)},\boldsymbol{x}\right)+\widetilde{\hat{w}_0}\right)$$

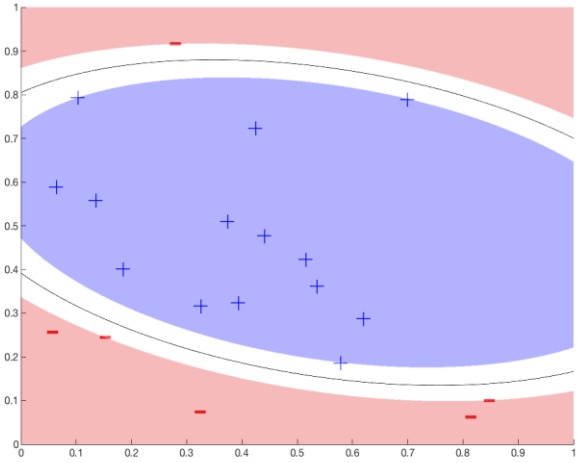


Smaller C





Larger C



Hard Margin

2^{nd}-Degree Polynomial Kernel

C is a trade-off parameter between the size of the margin and the soft training error

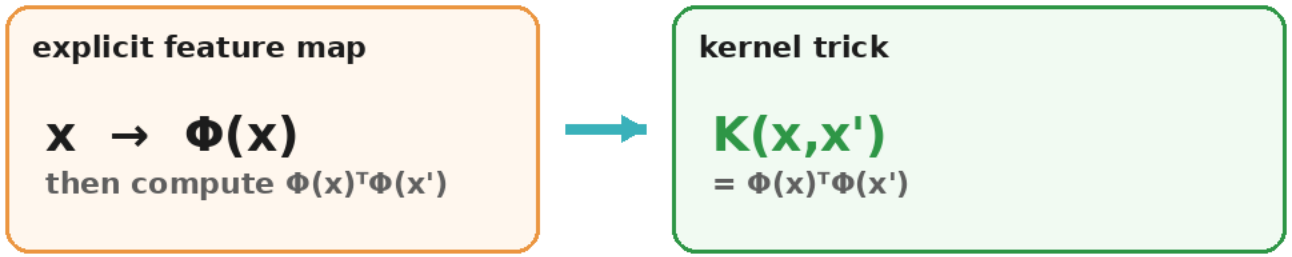
Myth Busters

MythBusters #1

“The kernel trick means there are no features anymore.”

Myth: kernels replace feature maps with pure algebraic magic.

Reality: a kernel is still an inner product in some transformed feature space $\Phi(x)$. The trick is that we compute it implicitly.



You keep the transformed-space geometry without writing down all transformed coordinates.

Valid Kernels

- Any function K is a valid kernel if and only if:
  - $\exists$ a transformation Φ s.t.

$$K(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}') \; \forall \; \boldsymbol{x}, \boldsymbol{x}'$$

$$\Updownarrow$$

  - the Gram matrix

$$\mathrm{K} = \begin{bmatrix} K(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & K(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) & \cdots & K(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(N)}) \\ K(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) & K(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(2)}) & \cdots & K(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(N)}) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}) & K(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(2)}) & \cdots & K(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(N)}) \end{bmatrix}$$

  is symmetric and positive semi-definite $\forall$ sets

  $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$

Recall: K is positive semi-definite iff $\mathbf{x}^T \mathrm{K} \mathbf{x} \geq 0$ for all $\mathbf{x}$

Building New Kernels

- For any valid kernels K_1, K_2 with implied feature transformations Φ_1, Φ_2 and non-negative coefficients c_1, c_2, the following are all valid kernels:

1. $K(\boldsymbol{x}, \boldsymbol{x}') = c_1 K_1(\boldsymbol{x}, \boldsymbol{x}') + c_2 K_2(\boldsymbol{x}, \boldsymbol{x}')$

   $\Phi(\boldsymbol{x}) = [\sqrt{c_1}\Phi_1(\boldsymbol{x}), \sqrt{c_1}\Phi_2(\boldsymbol{x})]$

2. $K(\boldsymbol{x}, \boldsymbol{x}') = c_1 K_1(\boldsymbol{x}, \boldsymbol{x}') K_2(\boldsymbol{x}, \boldsymbol{x}')$

   $$\Phi(\boldsymbol{x}) = \left[\left\{\sqrt{c_1}\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\right\}_{\phi_i(\boldsymbol{x})\in\Phi_1(\boldsymbol{x}),\phi_j(\boldsymbol{x})\in\Phi_2(\boldsymbol{x})}\right]$$

3. $K(\boldsymbol{x}, \boldsymbol{x}') = e^{K_1(\boldsymbol{x},\boldsymbol{x}')}$

Taylor series: $e^{K_1(\boldsymbol{x},\boldsymbol{x}')} = 1 + K_1(\boldsymbol{x}, \boldsymbol{x}') + \frac{K_1(\boldsymbol{x},\boldsymbol{x}')^2}{2!} + \frac{K_1(\boldsymbol{x},\boldsymbol{x}')^3}{3!} + \cdots$

Key Takeaways

- Kernels and the “kernel trick” allow for efficient use of feature transformations for inner product methods
  - Definition of valid kernels
  - Common kernels and combining kernels

Kernels Everywhere!

- Any method that only depends on the Euclidean distance between data points is an inner product method:

$$\|\boldsymbol{x} - \boldsymbol{x}'\|_2 = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T(\boldsymbol{x} - \boldsymbol{x}')} = \sqrt{\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{x}' + \boldsymbol{x}'^T\boldsymbol{x}'}$$

  - We can kernelize kNN!
- We can also kernelize logistic/linear/ridge regression!

Kernels in Logistic Regression

$$P(Y = 1 \mid x, \mathbf{w}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)}}$$

- Define weights in terms of features:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$$

$$P(Y = 1 \mid x, \mathbf{w}) = \frac{1}{1 + e^{-(\sum_i \alpha_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b)}} = \frac{1}{1 + e^{-(\sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b)}}$$

- Derive simple gradient descent rule on α_i

Ridge regression

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda \|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Recall

$$\mathbf{A} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1^{(1)} & \dots & X_1^{(p)} \\ \vdots & \ddots & \vdots \\ X_n^{(1)} & \dots & X_n^{(p)} \end{bmatrix}$$

Hence $\mathbf{A}^T\mathbf{A}$ is a p x p matrix whose entries denote the (sample) correlation between the features

NOT inner products between the data points – the inner product matrix would be $\mathbf{A}\mathbf{A}^T$ which is n x n (also known as Gram matrix)

Using dual formulation, we will write the solution in terms of $\mathbf{A}\mathbf{A}^T$

Ridge regression

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

<u>Similarity with SVMs</u>

Primal problem:

$$\min_{\beta, z_i} \sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|_2^2$$
$$\text{s.t. } z_i = Y_i - X_i\beta$$

SVM Primal problem:

$$\min_{w, \xi_i} C\sum_{i=1}^{n} \xi_i + \frac{1}{2}\|w\|_2^2$$
$$\text{s.t. } \xi_i = \max(1 - Y_i X_i \cdot w, 0)$$

Lagrangian:

$$\sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i(z_i - Y_i + X_i\beta)$$

α_i – Lagrange parameter, one per training point

Ridge regression (dual)

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Dual problem:

$$\max_{\alpha} \min_{\beta, z_i} \sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i(z_i - Y_i + X_i\beta)$$

$\alpha = \{\alpha_i\}$ for i = 1,…, n

Taking derivatives of Lagrangian wrt β and z_i we get:

$$\beta = -\frac{1}{2\lambda}\mathbf{A}^\top\alpha \qquad z_i = -\frac{\alpha_i}{2}$$

Dual problem: $$\max_{\alpha} \; -\frac{\alpha^\top\alpha}{4} - \frac{1}{4\lambda}\alpha^\top\mathbf{A}\mathbf{A}^\top\alpha - \alpha^\top\mathbf{Y}$$

n-dimensional optimization problem

Ridge regression (dual)

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

$$= \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

Dual problem:

$$\max_{\alpha} \; -\frac{\alpha^\top\alpha}{4} - \frac{1}{4\lambda}\alpha^\top\mathbf{A}\mathbf{A}^\top\alpha - \alpha^\top\mathbf{Y} \qquad \Rightarrow \widehat{\alpha} = -2\left(\frac{\mathbf{A}\mathbf{A}^\top}{\lambda} + \mathbf{I}\right)^{-1}\mathbf{Y}$$

can get back $\hat{\beta} = -\frac{1}{2\lambda}\mathbf{A}^\top\hat{\alpha} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top + \lambda\mathbf{I})^{-1}\mathbf{Y}$

Weighted average of training points

Weight of each training point (but typically not sparse)

Kernelized ridge regression

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Using dual, can re-write solution as:

$$\widehat{\beta} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

How does this help?

- Only need to invert n x n matrix (instead of p x p or m x m)
- More importantly, kernel trick!

  AA^T involves only inner products between the training points
  BUT still have an extra **A**^{T}

  Recall the predicted label is $\widehat{f}_n(X) = \mathbf{X}\widehat{\beta}$

  $$= \mathbf{X}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

  XA^T contains inner products between test point **X** and training points!

Kernelized ridge regression

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y} \qquad \widehat{f}_n(X) = \mathbf{X}\widehat{\beta}$$

Using dual, can re-write solution as:

$$\widehat{\beta} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

How does this help?

- Only need to invert n x n matrix (instead of p x p or m x m)
- More importantly, kernel trick!

$$\widehat{f}_n(X) = \mathbf{K}_X(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

$$\mathbf{K}_X(i) = \boldsymbol{\phi}(X) \cdot \boldsymbol{\phi}(X_i)$$
$$\mathbf{K}(i,j) = \boldsymbol{\phi}(X_i) \cdot \boldsymbol{\phi}(X_j)$$

Work with kernels, never need to write out the high-dim vectors

Kernelized ridge regression

$$\widehat{f}_n(X) = \mathbf{K}_X(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y}$$

$$\mathbf{K}_X(i) = \boldsymbol{\phi}(X) \cdot \boldsymbol{\phi}(X_i)$$
$$\mathbf{K}(i,j) = \boldsymbol{\phi}(X_i) \cdot \boldsymbol{\phi}(X_j)$$

Work with kernels, never need to write out the high-dim vectors

Examples of kernels:

Polynomials of degree exactly d $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$

Polynomials of degree up to d $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$

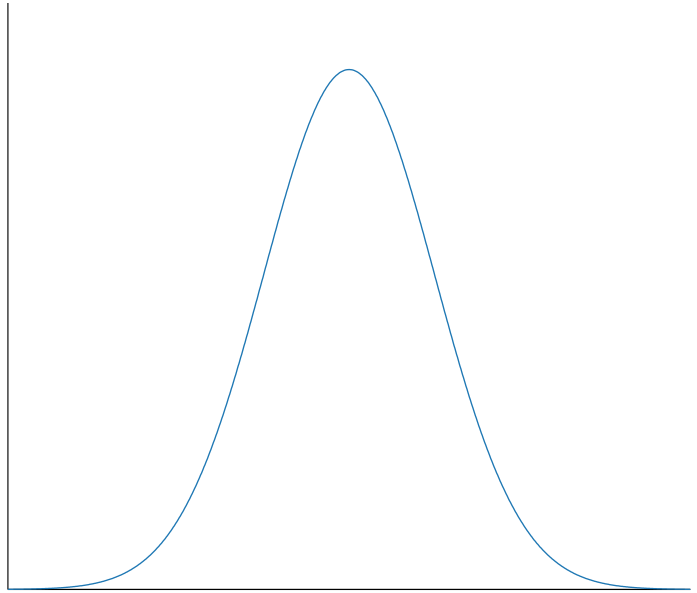
Gaussian/Radial kernels $K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{2\sigma^2}\right)$

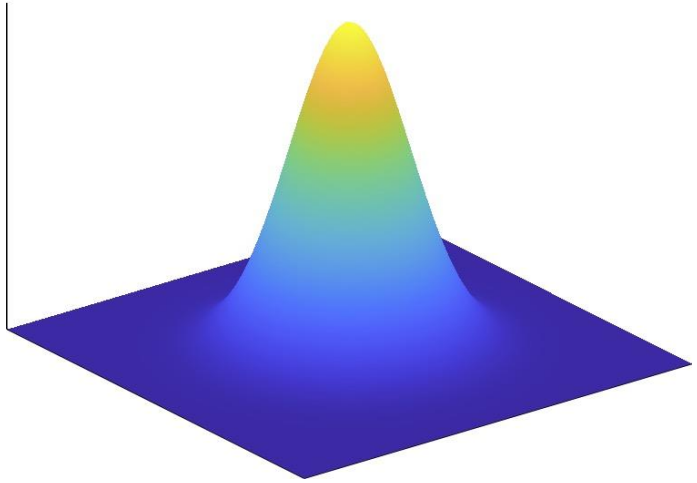
Ridge Regression with (implicit) nonlinear features $\boldsymbol{\phi}(X)$! $f(X) = \phi(X)\beta$

Gaussians

(Univariate) Gaussians:

$$x \sim \mathcal{N}(x; \mu = 0, \sigma^2 = 1)$$





- Multivariate Gaussians:

$$\boldsymbol{x} = [x_1, \ldots, x_D]^T$$

$$\sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu} = \mathbf{0}_D, \Sigma = I_D)$$

Some fun facts about Gaussians

- Closure under linear transformations:

$$\text{If } \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma),$$

$$\text{then } A\boldsymbol{x} + b \sim \mathcal{N}(A\boldsymbol{\mu} + b, A\Sigma A^T)$$

- Closure under addition

$$\text{If } \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) \text{ and } \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}; \boldsymbol{m}, S),$$

$$\text{then } \boldsymbol{x} + \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{m}, \Sigma + S)$$

- Closure under conditioning:

$$\text{If } \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right),$$

$$\text{then } x_1 | x_2 = c \sim \mathcal{N}(x_1; \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(c - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})$$

Some old friends

Gaussian process =

Bayesian linear regression + Kernels

Some old friends

Gaussian process =

Bayesian linear regression + Kernels

Recall: MAP for Linear Regression

- If we assume a linear model with additive Gaussian noise

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \rightarrow \boldsymbol{y} \sim N(X\boldsymbol{w}, \sigma^2 I_N)$$

and independent identical Gaussian priors on the weights…

$$\boldsymbol{w} \sim N\left(\mathbf{0} \ , \frac{\sigma^2}{\lambda} I_{D+1}\right) \rightarrow p(\boldsymbol{w}) \propto \exp\left(-\frac{1}{2\sigma^2}(\lambda \boldsymbol{w}^T \boldsymbol{w})\right)$$

- … then, the MAP of $\boldsymbol{w}$ is the ridge regression solution!

$$\begin{aligned} \boldsymbol{w}_{MAP} &= \underset{\boldsymbol{w}}{\text{argmin}} \, (X\boldsymbol{w} - \boldsymbol{y})^T (X\boldsymbol{w} - \boldsymbol{y}) + \lambda \boldsymbol{w}^T \boldsymbol{w} \\ &= (X^T X + \lambda I_{D+1})^{-1} X^T \boldsymbol{y} \end{aligned}$$

Bayesian Linear Regression

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{w} \sim N(\mathbf{0}_{D+1}, \Sigma)$$

then,

$$\boldsymbol{y} \sim N((X\mathbf{0}_{D+1} + \mathbf{0}_N) \equiv \mathbf{0}_N, X\Sigma X^T + \sigma^2 I_N)$$

Bayesian Linear Regression

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{w} \sim N(\mathbf{0}_{D+1}, \Sigma)$$

then,

$$\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{y} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0}_{D+1} \\ \mathbf{0}_N \end{bmatrix}, \begin{bmatrix} \Sigma & ??? \\ ??? & X\Sigma X^T + \sigma^2 I_N \end{bmatrix}\right)$$

- Covariance between $\boldsymbol{y}$ and $\boldsymbol{w}$:

$$\mathrm{Cov}(\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon}, \boldsymbol{w}) = \mathrm{Cov}(X\boldsymbol{w}, \boldsymbol{w}) = X\mathrm{Cov}(\boldsymbol{w}, \boldsymbol{w}) = X\Sigma$$

Bayesian Linear Regression

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{w} \sim N(\mathbf{0}_{D+1}, \Sigma)$$

then,

$$\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{y} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0}_{D+1} \\ \mathbf{0}_N \end{bmatrix}, \begin{bmatrix} \Sigma & X\Sigma \\ X\Sigma & X\Sigma X^T + \sigma^2 I_N \end{bmatrix}\right)$$

- Covariance between $\boldsymbol{y}$ and $\boldsymbol{w}$:

$$\mathrm{Cov}(\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon}, \boldsymbol{w}) = \mathrm{Cov}(X\boldsymbol{w}, \boldsymbol{w}) = X\mathrm{Cov}(\boldsymbol{w}, \boldsymbol{w}) = X\Sigma$$

Bayesian Linear Regression

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{w} \sim N(\mathbf{0}_{D+1}, \Sigma)$$

then,

$$\boldsymbol{w} \mid \boldsymbol{y} \sim N(\boldsymbol{\mu}_{POST}, \Sigma_{POST})$$

where

$$\boldsymbol{\mu}_{POST} = \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} \boldsymbol{y},$$
$$\Sigma_{POST} = \Sigma - \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} X\Sigma$$

Bayesian Linear Regression

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{w} \sim N(\mathbf{0}_{D+1}, \Sigma)$$

then given a new test data point $\boldsymbol{x}'$, the prediction is

$$y' \mid \boldsymbol{y} = \boldsymbol{x}'^T \boldsymbol{w} \mid \boldsymbol{y} \sim N\left(\boldsymbol{x}'^T \boldsymbol{\mu}_{POST}, \boldsymbol{x}'^T \Sigma_{POST} \boldsymbol{x}'\right)$$

where

$$\boldsymbol{\mu}_{POST} = \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} \boldsymbol{y},$$
$$\Sigma_{POST} = \Sigma - \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} X\Sigma$$

Bayesian Linear Regression

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{w} \sim N(\mathbf{0}_{D+1}, \Sigma)$$

then given a new test data point $\boldsymbol{x}'$, the prediction is

$$y' \mid \boldsymbol{y} = \boldsymbol{x}'^T \boldsymbol{w} \mid \boldsymbol{y} \sim N(\boldsymbol{\mu}_{PRED}, \Sigma_{PRED})$$

where

$$\boldsymbol{\mu}_{PRED} = \boldsymbol{x}'^T \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} \boldsymbol{y},$$

$$\Sigma_{PRED} = \boldsymbol{x}'^T \Sigma \boldsymbol{x}' - \boldsymbol{x}'^T \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} X\Sigma \boldsymbol{x}'$$

Some old friends

Gaussian process =

Bayesian linear regression + Kernels

Some old friends

Gaussian process =

Bayesian linear regression + Kernels

Bayesian Linear Regression...

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{w} \sim N(\mathbf{0}_{D+1}, \Sigma)$$

then given a new test data point $\boldsymbol{x}'$, the prediction is

$$y' \mid \boldsymbol{y} = \boldsymbol{x}'^T \boldsymbol{w} \mid \boldsymbol{y} \sim N(\boldsymbol{\mu}_{PRED}, \Sigma_{PRED})$$

where

$$\boldsymbol{\mu}_{PRED} = \boldsymbol{x}'^T \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} \boldsymbol{y},$$

$$\Sigma_{PRED} = \boldsymbol{x}'^T \Sigma \boldsymbol{x}' - \boldsymbol{x}'^{\mathrm{T}} \Sigma X^T (X\Sigma X^T + \sigma^2 I_N)^{-1} X\Sigma \boldsymbol{x}'$$

Bayesian Linear Regression can be kernelized!

$$\Phi = \begin{bmatrix} 1 & \phi(\boldsymbol{x}^{(1)})^T \\ 1 & \phi(\boldsymbol{x}^{(2)})^T \\ \vdots & \vdots \\ 1 & \phi(\boldsymbol{x}^{(N)})^T \end{bmatrix}$$

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = \Phi\boldsymbol{\omega} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{\omega} \sim N(\mathbf{0}_{D'+1}, \Sigma)$$

then given a new test data point $\boldsymbol{x}'$, the prediction is

$$y' \mid \boldsymbol{y} = \phi(\boldsymbol{x}')^T\boldsymbol{\omega} \mid \boldsymbol{y} \sim N(\boldsymbol{\mu}_{PRED}, \Sigma_{PRED})$$

where

$$\boldsymbol{\mu}_{PRED} = \phi(\boldsymbol{x}')^T\Sigma\Phi^T(\Phi\Sigma\Phi^T + \sigma^2 I_N)^{-1}\boldsymbol{y},$$

$$\Sigma_{PRED} = \phi(\boldsymbol{x}')^T\Sigma\phi(\boldsymbol{x}') - \phi(\boldsymbol{x}')^T\Sigma\Phi^T(\Phi\Sigma\Phi^T + \sigma^2 I_N)^{-1}\Phi\Sigma\phi(\boldsymbol{x}')$$

Bayesian Linear Regression can be kernelized!

$$\Phi = \begin{bmatrix} 1 & \phi(\boldsymbol{x}^{(1)})^T \\ 1 & \phi(\boldsymbol{x}^{(2)})^T \\ \vdots & \vdots \\ 1 & \phi(\boldsymbol{x}^{(N)})^T \end{bmatrix}$$

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = \Phi\boldsymbol{\omega} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{\omega} \sim N(\mathbf{0}_{D'+1}, \Sigma)$$

then given a new test data point $\boldsymbol{x}'$, the prediction is

$$y' \mid \boldsymbol{y} = \phi(\boldsymbol{x}')^T\boldsymbol{\omega} \mid \boldsymbol{y} \sim N(\boldsymbol{\mu}_{PRED}, \Sigma_{PRED})$$

where

$$\boldsymbol{\mu}_{PRED} = \phi(\boldsymbol{x}')^T\Sigma\Phi^T(\Phi\Sigma\Phi^T + \sigma^2 I_N)^{-1}\boldsymbol{y},$$

$$\Sigma_{PRED} = \phi(\boldsymbol{x}')^T\Sigma\phi(\boldsymbol{x}') - \phi(\boldsymbol{x}')^T\Sigma\Phi^T(\Phi\Sigma\Phi^T + \sigma^2 I_N)^{-1}\Phi\Sigma\phi(\boldsymbol{x}')$$

- Define the kernel function to be

$$K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T\Sigma\phi(\boldsymbol{x}')$$

Bayesian Linear Regression can be kernelized!

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = \Phi\boldsymbol{\omega} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{\omega} \sim N(\mathbf{0}_{D'+1}, \Sigma)$$

then given a new test data point $\boldsymbol{x}'$, the prediction is

$$y' \mid \boldsymbol{y} = \phi(\boldsymbol{x}')^T\boldsymbol{\omega} \mid \boldsymbol{y} \sim N(\boldsymbol{\mu}_{PRED}, \Sigma_{PRED})$$

where

$$\boldsymbol{\mu}_{PRED} = K(\boldsymbol{x}', X)(K(X, X) + \sigma^2 I_N)^{-1}\boldsymbol{y},$$

$$\Sigma_{PRED} = K(\boldsymbol{x}', \boldsymbol{x}') - K(\boldsymbol{x}', X)(K(X, X) + \sigma^2 I_N)^{-1}K(X, \boldsymbol{x}')$$

- Define the kernel function to be

$$K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T\Sigma\phi(\boldsymbol{x}')$$

Wait, what happened to the weights?

Recall: models as functions!

- Assume a linear model with additive Gaussian noise and a zero-mean Gaussian prior on the weights:

$$\boldsymbol{y} = \Phi\boldsymbol{\omega} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{\epsilon} \sim N(\mathbf{0}_N, \sigma^2 I_N) \text{ and } \boldsymbol{\omega} \sim N(\mathbf{0}_{D'+1}, \Sigma)$$

then given a new test data point $\boldsymbol{x}'$, the prediction is

$$y' \mid \boldsymbol{y} = \phi(\boldsymbol{x}')^T\boldsymbol{\omega} \mid \boldsymbol{y} \sim N(\boldsymbol{\mu}_{PRED}, \Sigma_{PRED})$$

where

$$\boldsymbol{\mu}_{PRED} = K(\boldsymbol{x}', X)(K(X, X) + \sigma^2 I_N)^{-1}\boldsymbol{y},$$

$$\Sigma_{PRED} = K(\boldsymbol{x}', \boldsymbol{x}') - K(\boldsymbol{x}', X)(K(X, X) + \sigma^2 I_N)^{-1}K(X, \boldsymbol{x}')$$

- Define the kernel function to be

$$K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T\Sigma\phi(\boldsymbol{x}')$$

Some old friends

Gaussian process =

Bayesian linear regression + Kernels

A new perspective

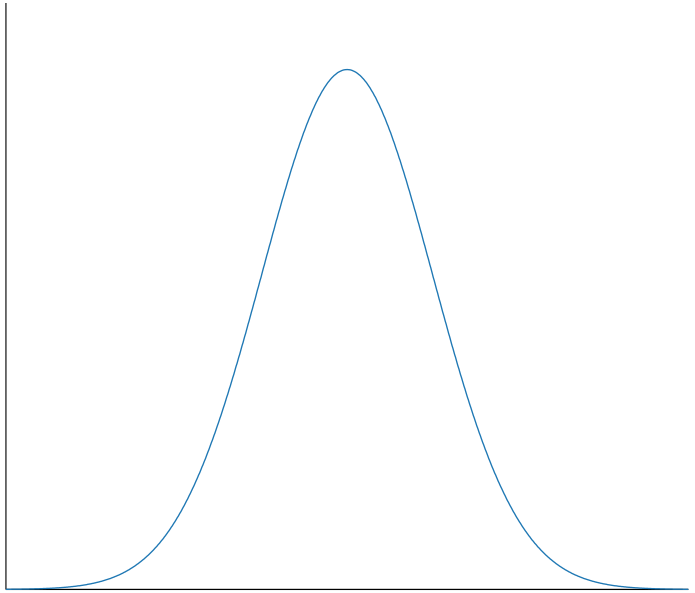
Gaussian process =

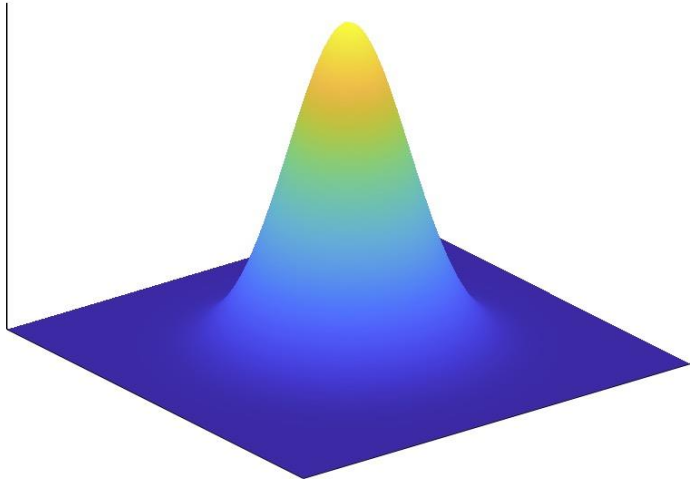
The extension of a Gaussian distribution to functions

Gaussians

- (Univariate) Gaussians:

$$x \sim \mathcal{N}(x; \mu = 0, \sigma^2 = 1)$$





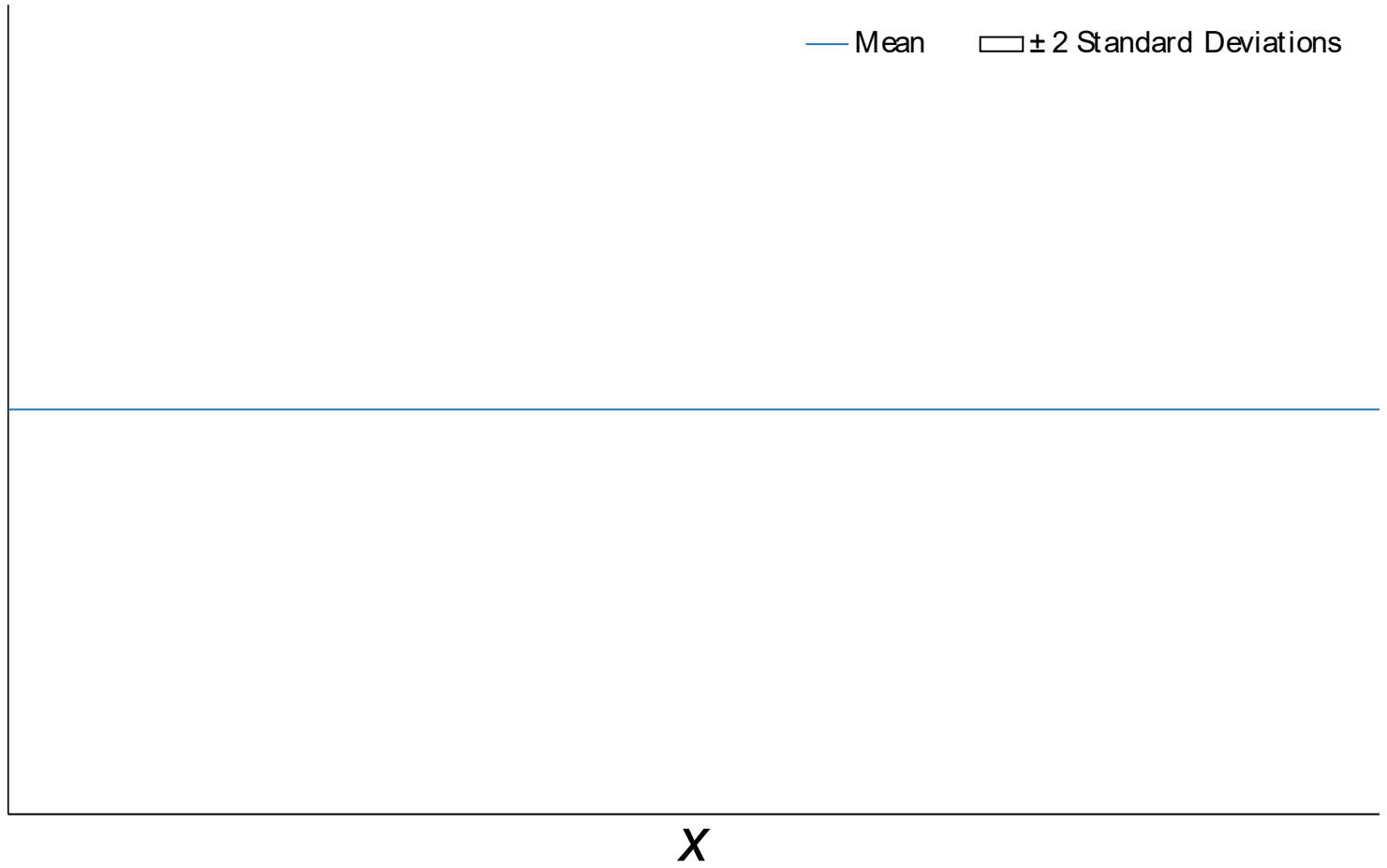
- Multivariate Gaussians:

$$\boldsymbol{x} = [x_1, \ldots, x_D]^T$$

$$\sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu} = \mathbf{0}_D, \Sigma = I_D)$$

Gaussian Process (GP)

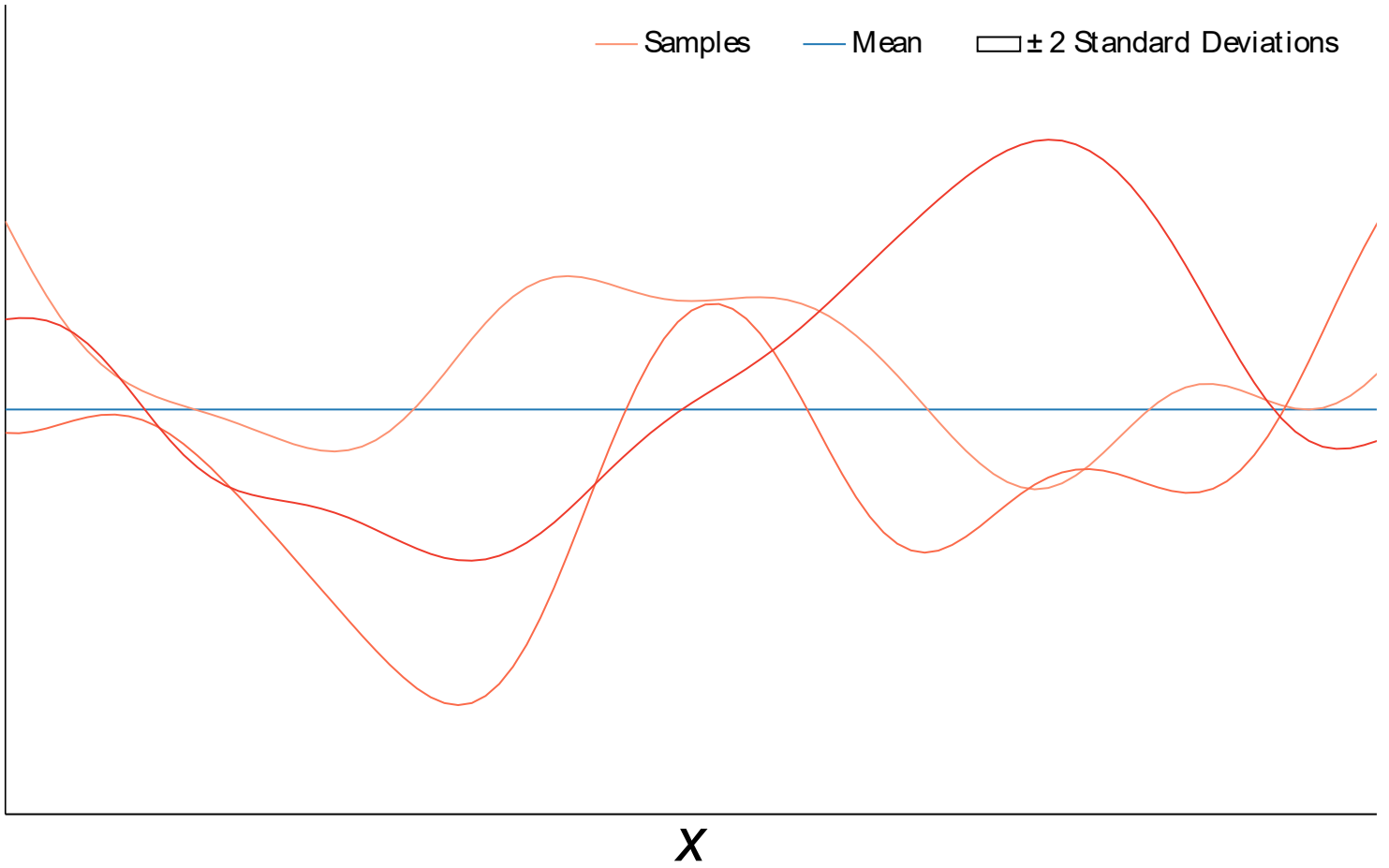
$$f: \mathbb{R}^p \mapsto \mathbb{R} \sim \mathcal{GP}(f; \mu(x) \quad , \Sigma(x, x')$$



$$f \sim \mathcal{GP}(\mu, \Sigma) \rightarrow f(x) \sim \mathcal{N}\big(\mu(x), \Sigma(x, x)\big)$$

Gaussian Process (GP)

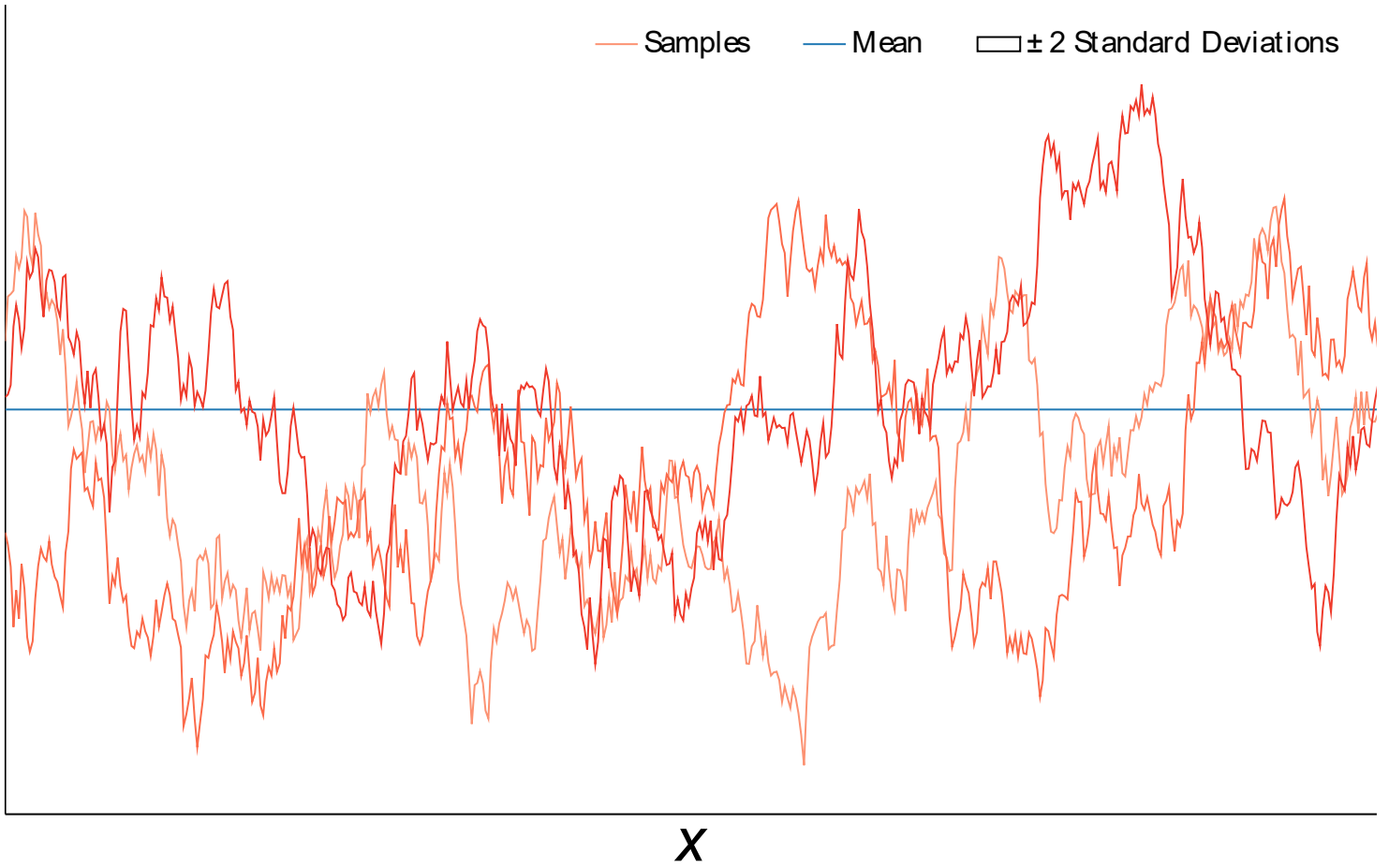
$$f: \mathbb{R}^p \mapsto \mathbb{R} \sim \mathcal{GP}(f; \mu(x) = 0, \Sigma(x, x') = \exp(-(x - x')^2))$$



$$f \sim \mathcal{GP}(\mu, \Sigma) \rightarrow f(x) \sim \mathcal{N}\big(\mu(x), \Sigma(x, x)\big)$$

Gaussian Process (GP)

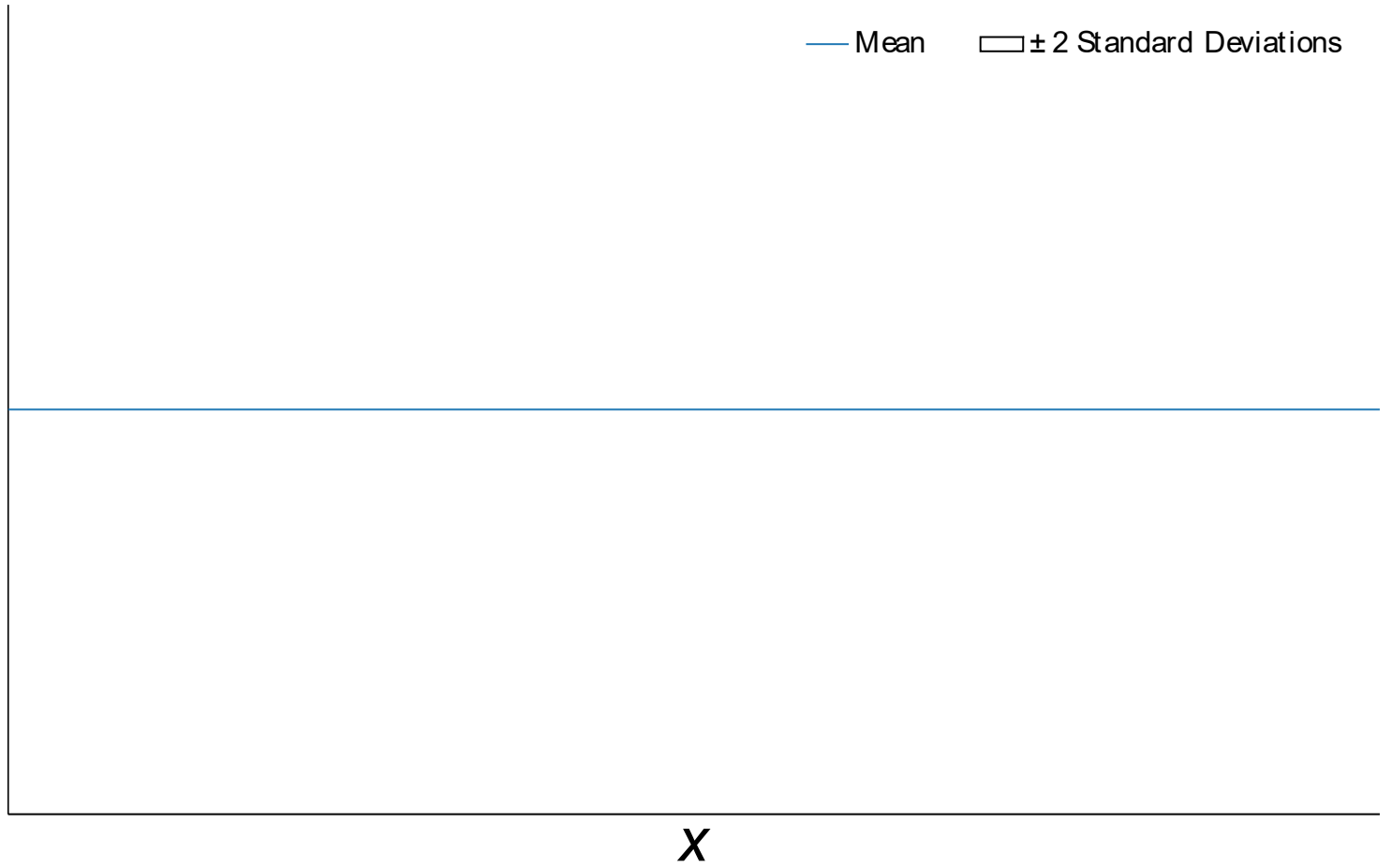
$$f: \mathbb{R}^p \mapsto \mathbb{R} \sim \mathcal{GP}(f; \mu(x) = 0, \Sigma(x, x') = \exp(-|x - x'|))$$



$$f \sim \mathcal{GP}(\mu, \Sigma) \rightarrow f(x) \sim \mathcal{N}\big(\mu(x), \Sigma(x, x)\big)$$

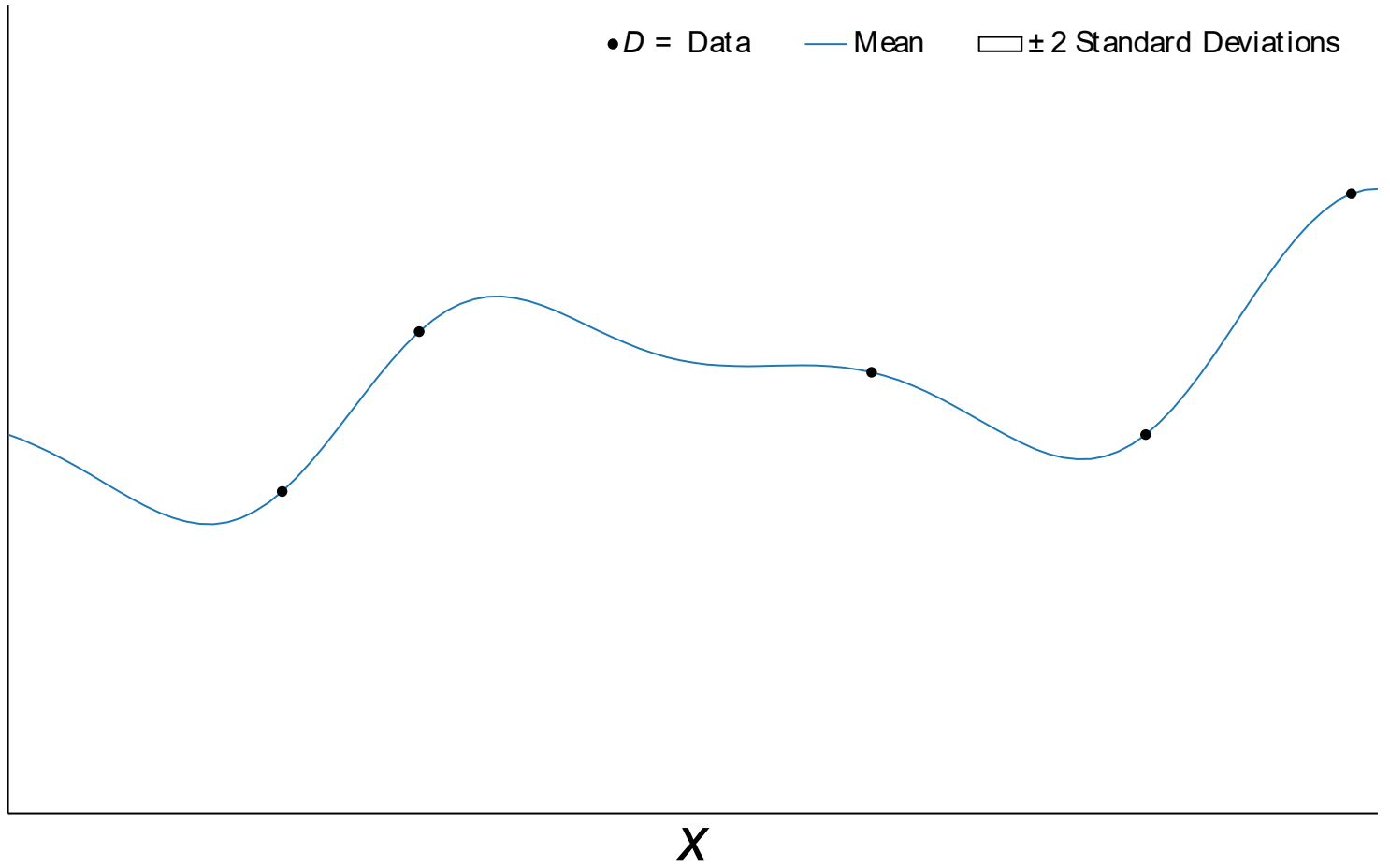
GP Prior

$$f: \mathbb{R}^p \mapsto \mathbb{R} \sim \mathcal{GP}(f; \mu(x) = 0, \Sigma(x, x') = \exp(-(x - x')^2))$$



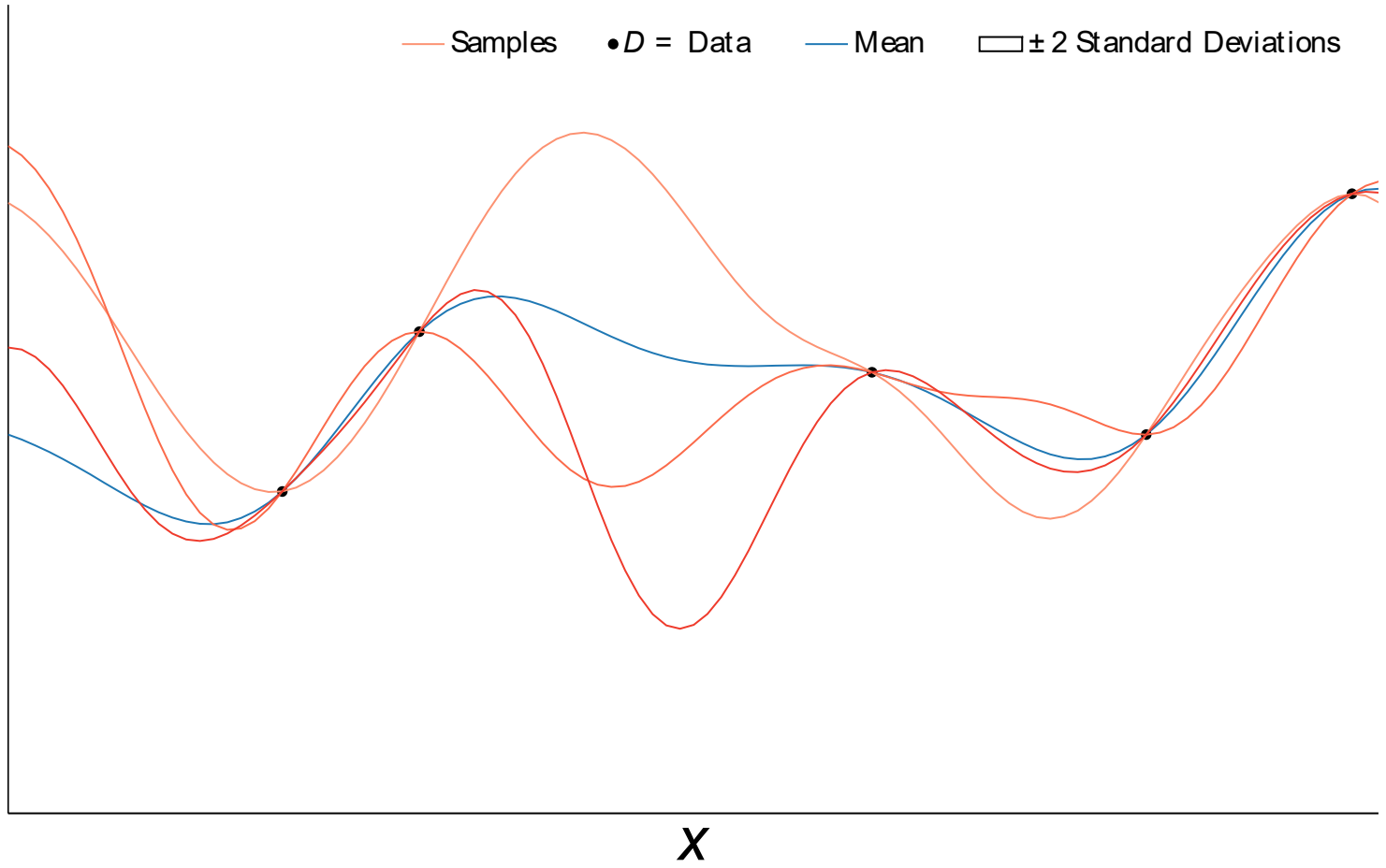
GP Posterior

$$f \mid \mathcal{D} \sim \mathcal{GP}(f; \mu_{\mathcal{D}}, \Sigma_{\mathcal{D}})$$



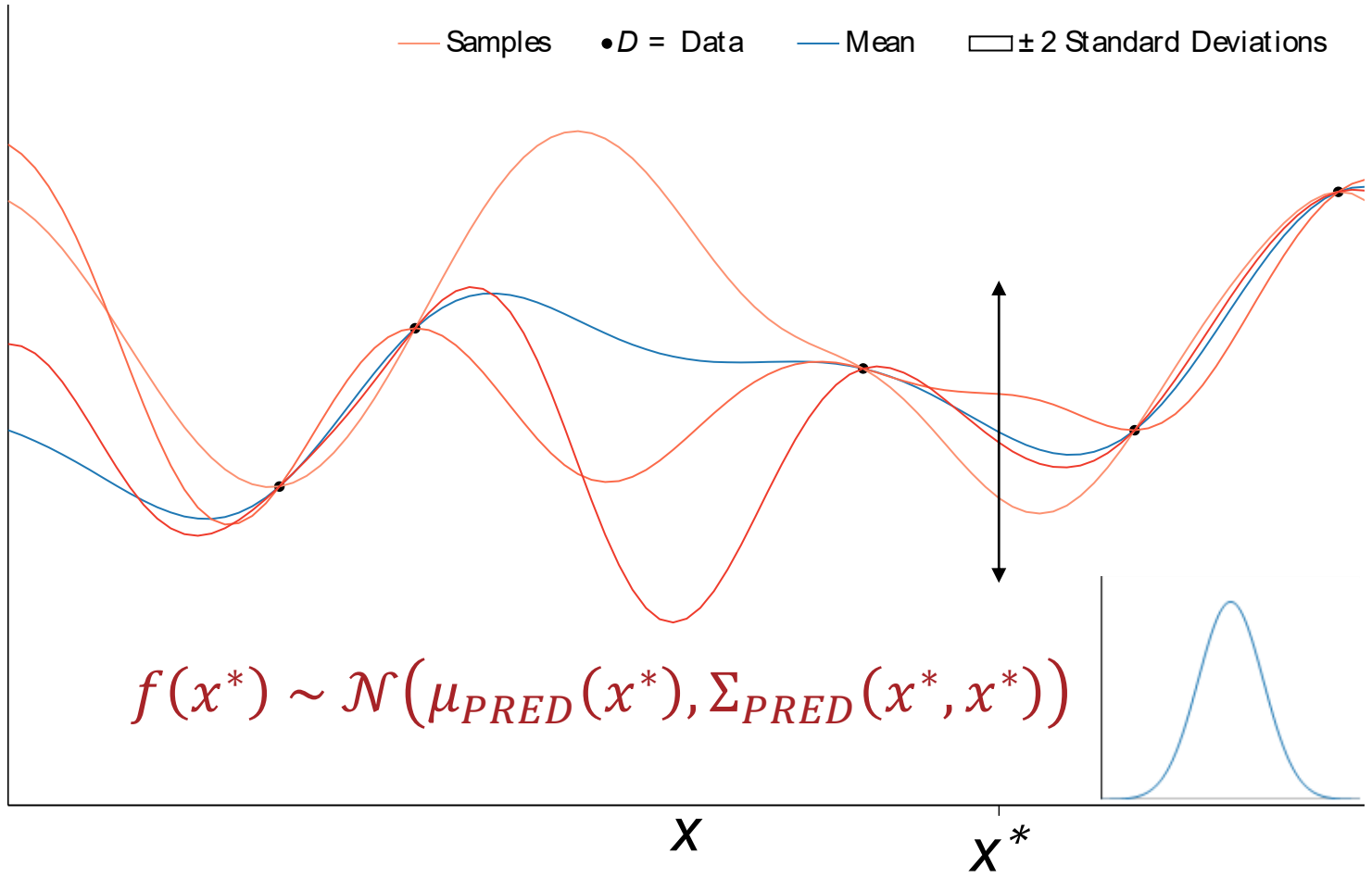
GP Posterior

$$f \mid \mathcal{D} \sim \mathcal{GP}(f; \mu_{\mathcal{D}}, \Sigma_{\mathcal{D}})$$



GP Posterior

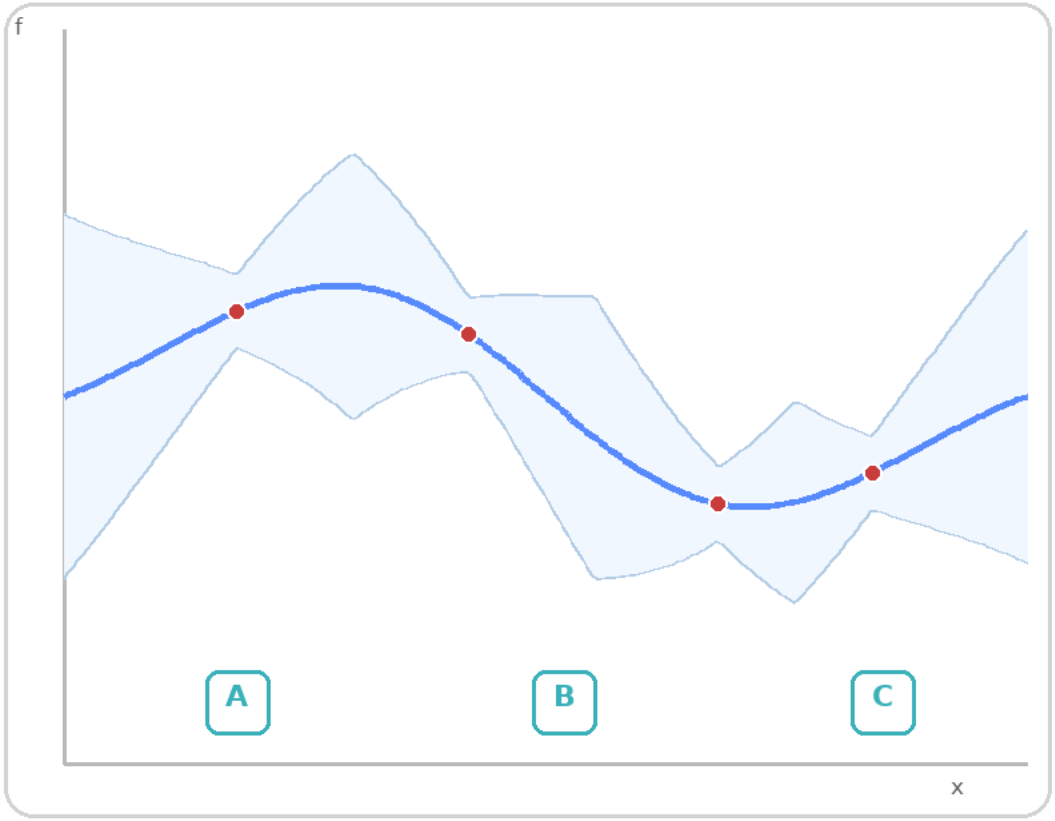
$$f \mid \mathcal{D} \sim \mathcal{GP}(f; \mu_{\mathcal{D}}, \Sigma_{\mathcal{D}})$$



Quick Game

Quick game: where is GP uncertainty smallest?

Look at the posterior picture intuition: where should the shaded confidence band be narrowest?



Vote

A. near an observed training point

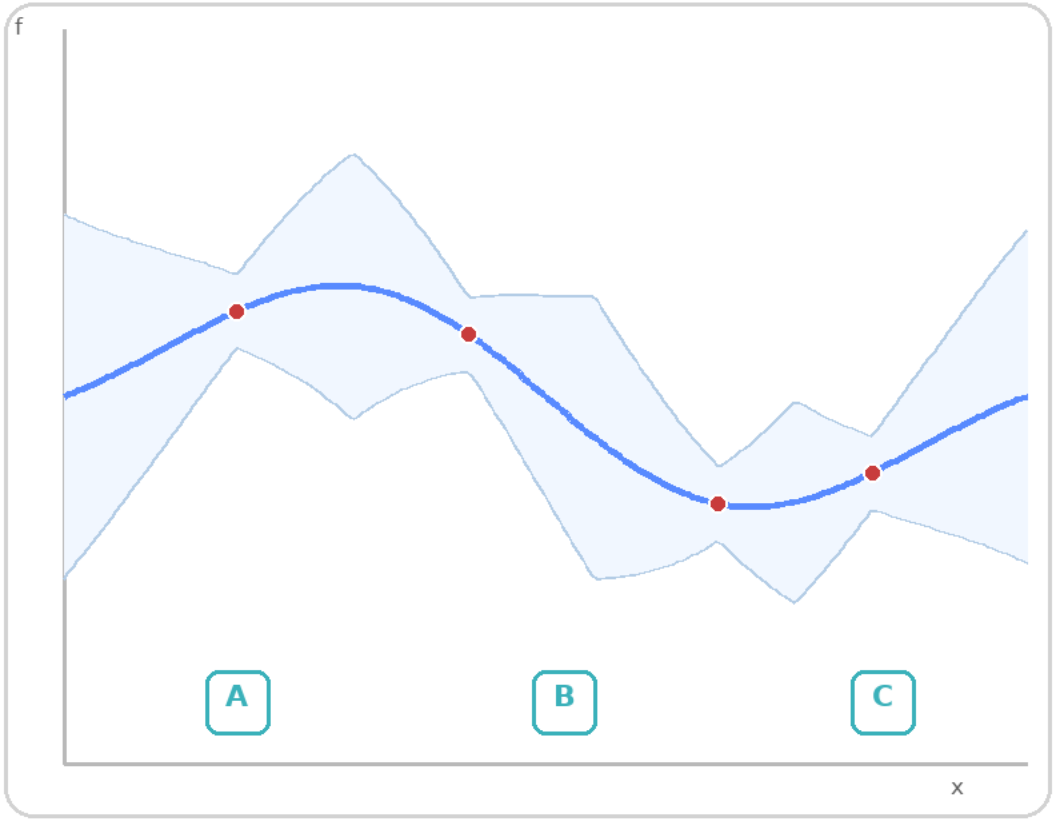
B. midway between observed points

C. far away from observed points

Quick Game

Quick game: where is GP uncertainty smallest?

Look at the posterior picture intuition: where should the shaded confidence band be narrowest?



Vote

A. near an observed training point

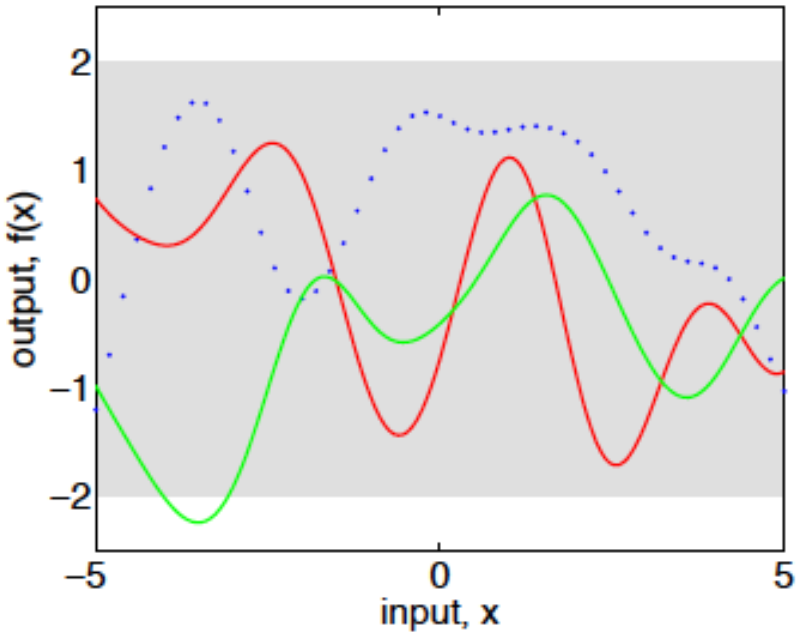
B. midway between observed points

C. far away from observed points

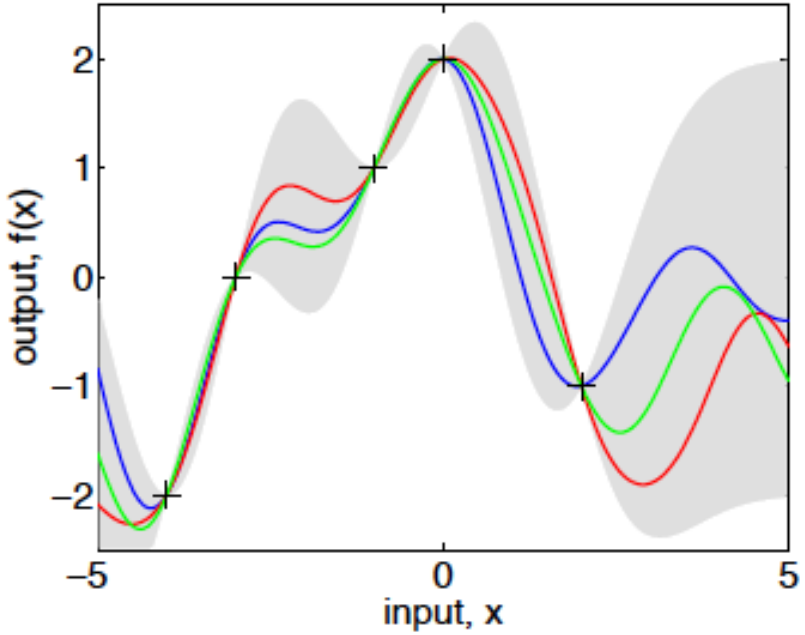
Reveal: A. In the GP posterior picture, uncertainty shrinks near observed points and grows away from them.

Gaussian process regression

3 functions drawn at random from GP **prior**



3 functions drawn at random from **posterior** based on five noiseless observations



Shaded region represents pointwise mean +/- 2 standard deviation

http://www.gaussianprocess.org/gpml/

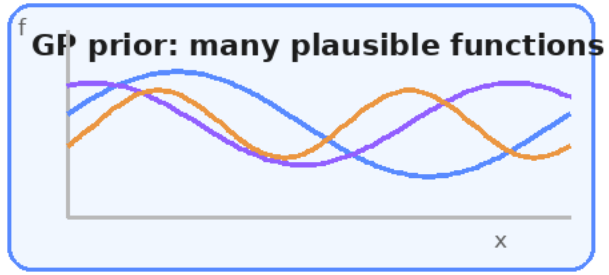
Myth Busters

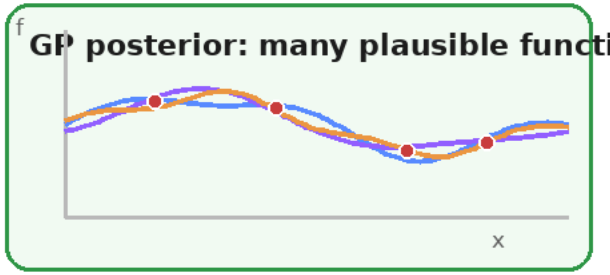
MythBusters #2

“A Gaussian process is just one very smooth fitted curve.”

Myth: a GP is just a single nonlinear regressor with uncertainty added later.

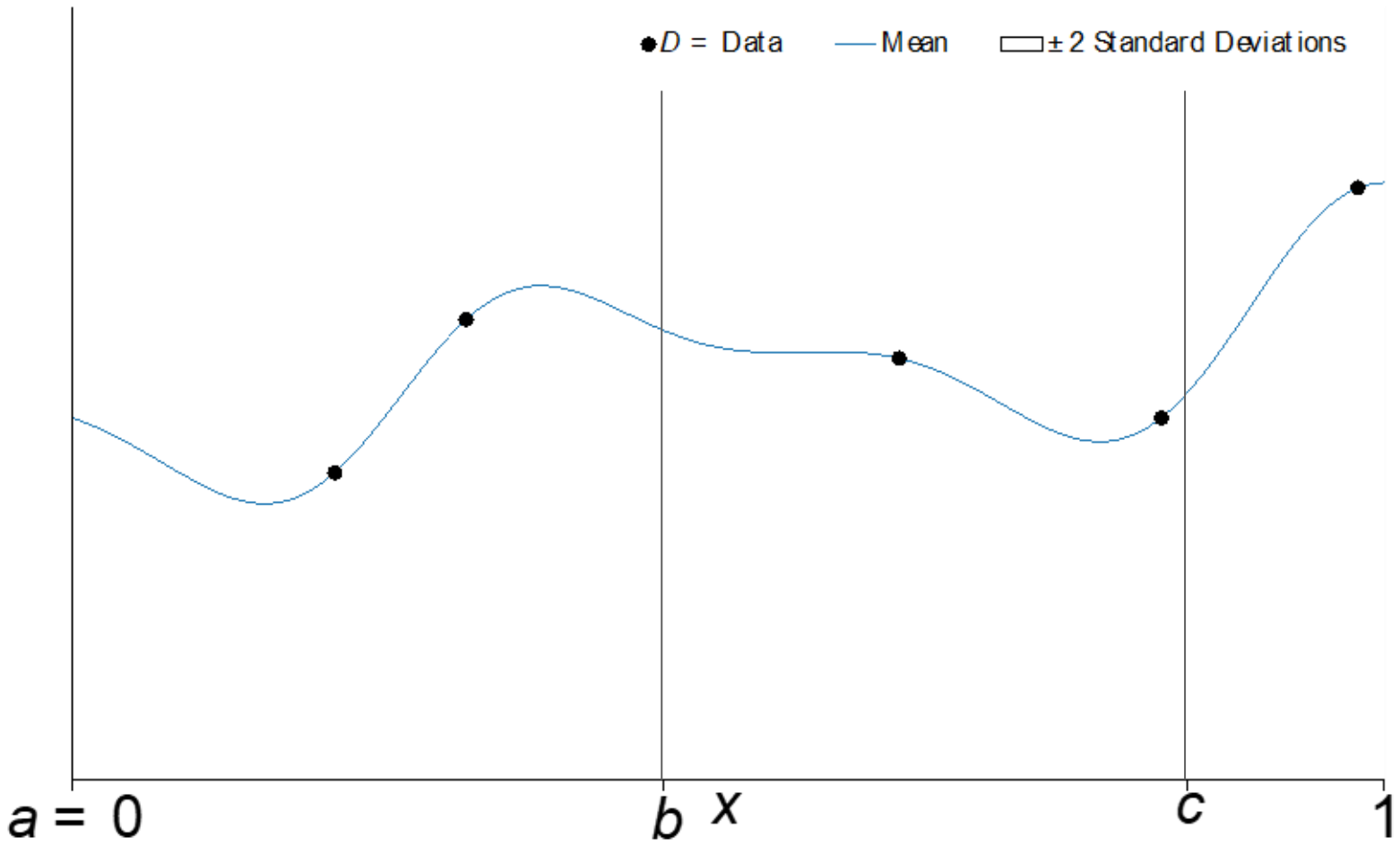
Reality: the lecture frames a GP as a distribution over functions. Data turns a prior over functions into a posterior over functions.





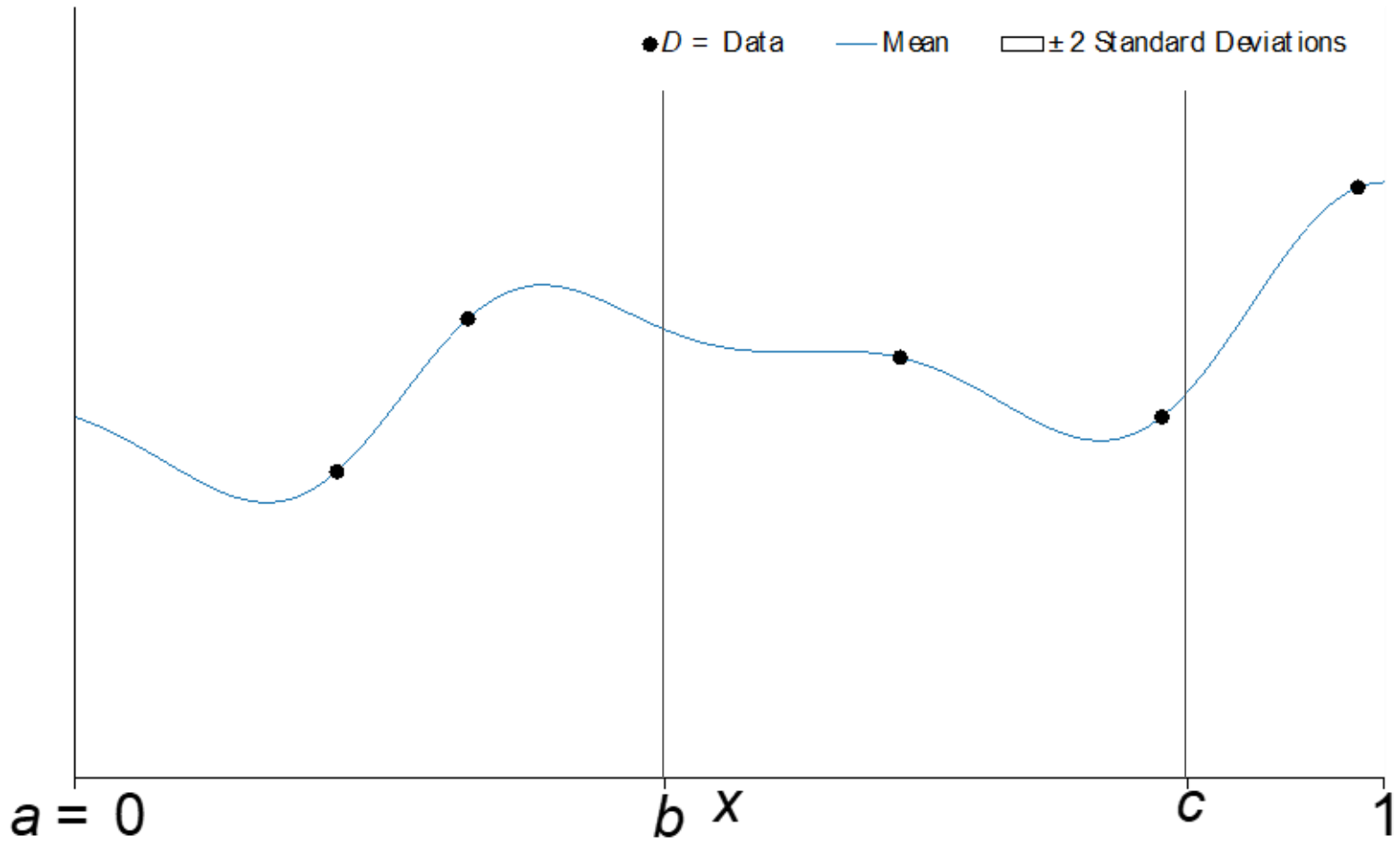
The mean is just one summary; the GP itself is the full distribution over functions.

Active Learning



Suppose you can add one data point to your training data.

Which value of x would you add and why?



Quick Game

Two truths and a lie (kernels & GPs)

Vote for the lie: A, B, or C.

A. The Gaussian-RBF kernel corresponds to an infinite-dimensional feature map.

B. Gaussian process regression can be viewed as Bayesian linear regression written in kernel form.

C. In the GP posterior picture, predictive variance is largest exactly at the observed noiseless training points.

Quick Game

Two truths and a lie (kernels & GPs)

Vote for the lie: A, B, or C.

A. The Gaussian-RBF kernel corresponds to an infinite-dimensional feature map.

B. Gaussian process regression can be viewed as Bayesian linear regression written in kernel form.

C. In the GP posterior picture, predictive variance is largest exactly at the observed noiseless training points.

Reveal: C is the lie.

Final Poll

Poll:

Lecture 25

Final poll (pick one)

Which statement is FALSE according to the lecture?

A. The kernel trick lets us compute inner products in transformed feature spaces implicitly.

B. A Gaussian process can be interpreted as a distribution over functions.

C. In a GP posterior, uncertainty is usually smallest near observed noiseless training points.

D. A smaller Gaussian-RBF radius r makes the kernel more local.

E. Gaussian process regression cannot be related to Bayesian linear regression.