AIMS CDT - Signal Processing

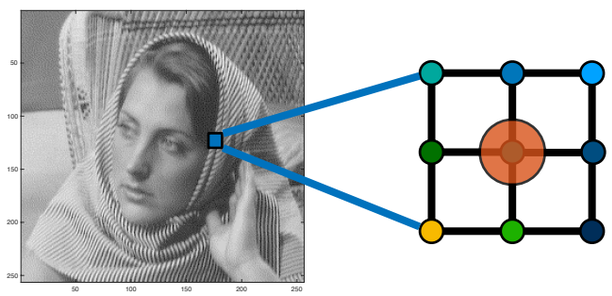
Michaelmas Term 2025

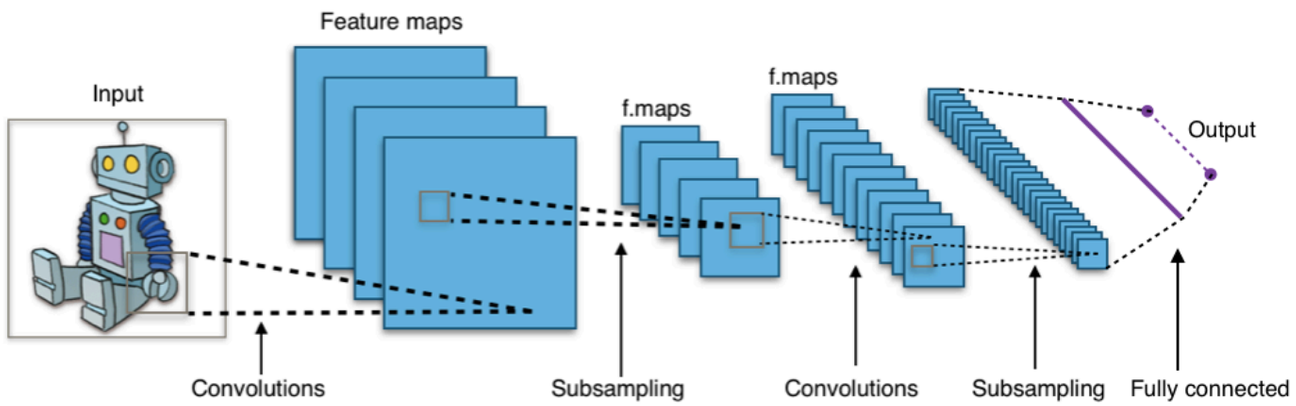
Xiaowen Dong

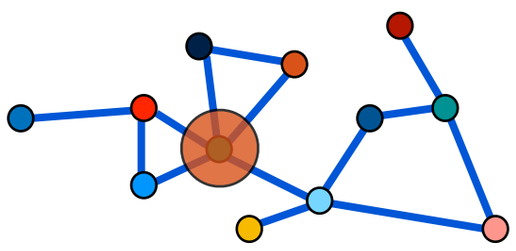
Department of Engineering Science

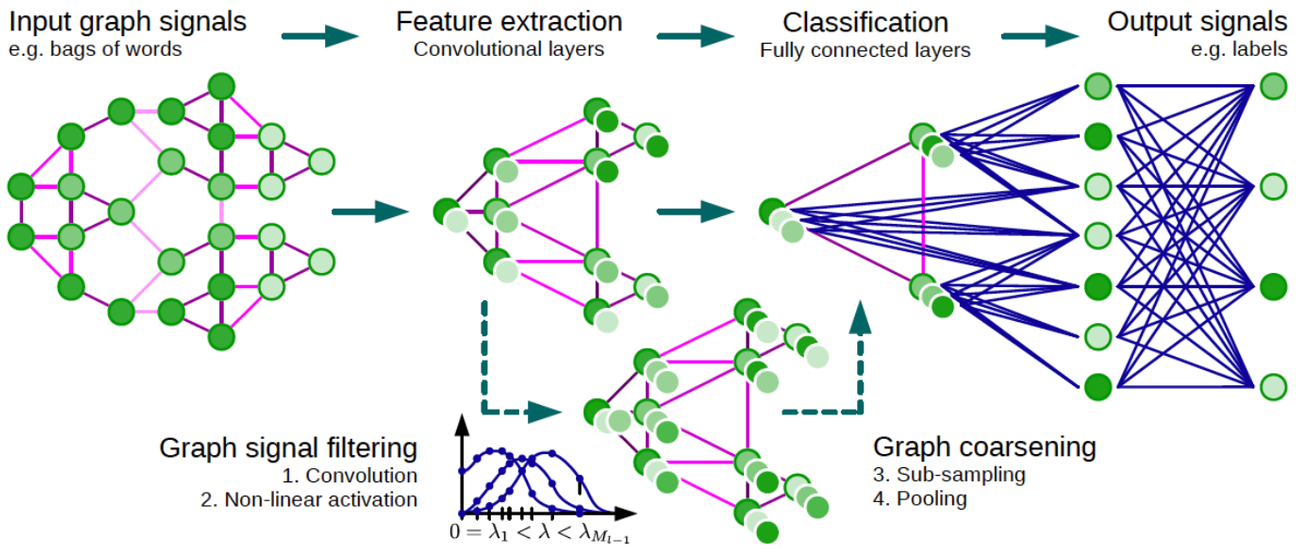
Bayesian Modelling of Graph-Structured Data

Deep learning on graphs



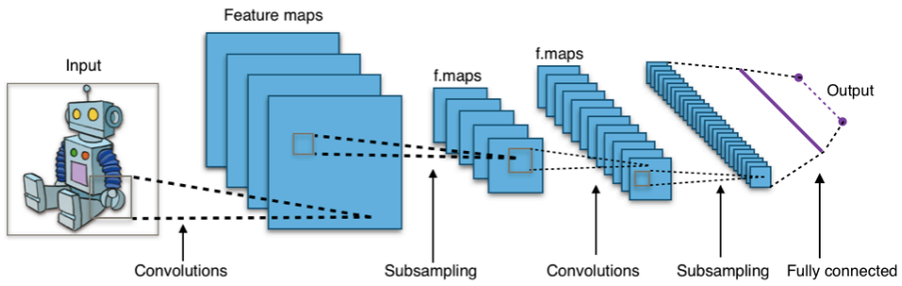




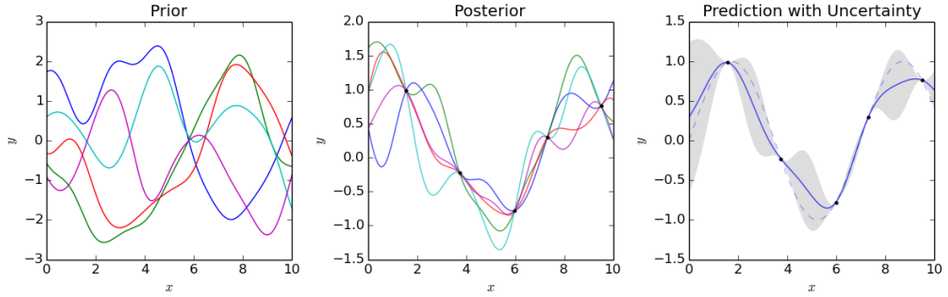


Deep learning vs Bayesian modelling

- Limitations of deep learning
  - do not usually incorporate prior knowledge
  - need large amount of training samples
  - produce point estimate without uncertainty



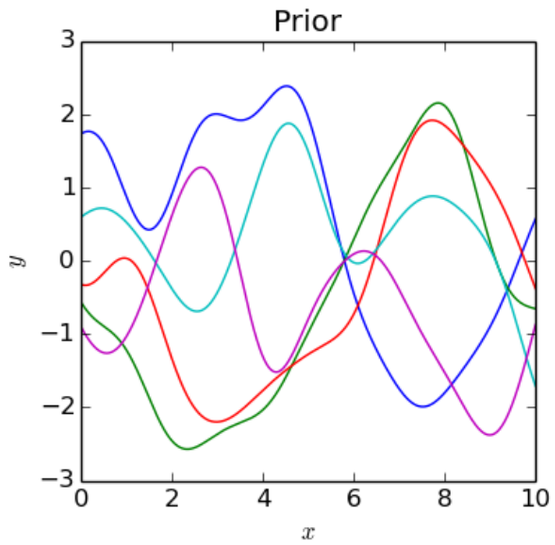
- Advantages of Bayesian modelling
  - incorporate prior knowledge about data
  - can be more sample efficient
  - produce uncertainty



Lecture 4

- Gaussian processes on graphs
- Bayesian optimisation of graph-based functions
- Discussion

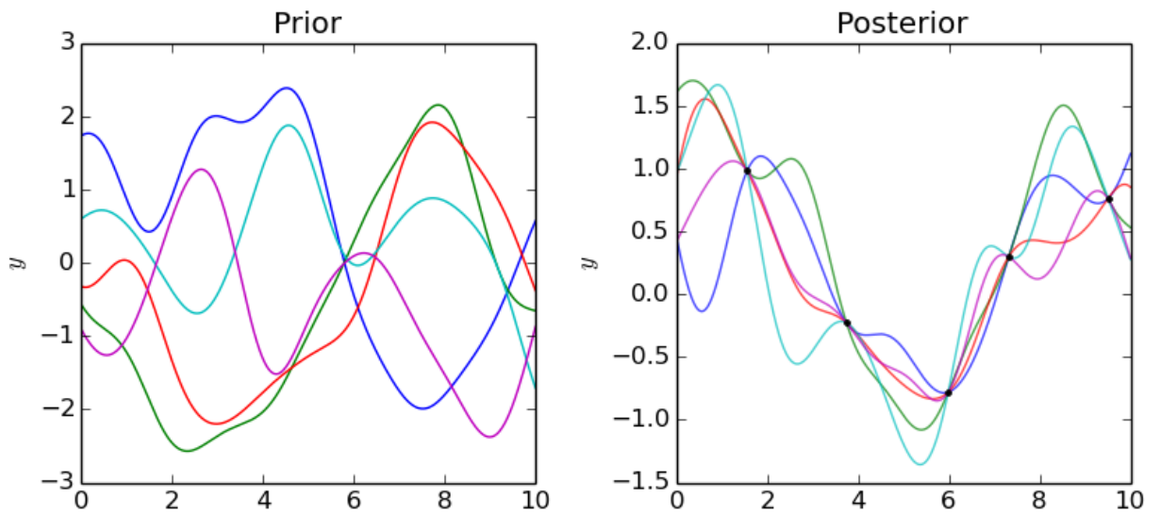
Gaussian processes



mean — covariance (kernel)

prior: $f(x) \sim \mathcal{GP}(\mu(x),\ k(x, x'))$

Gaussian processes



mean — covariance (kernel)

prior: $f(x) \sim \mathcal{GP}(\mu(x),\ k(x, x'))$

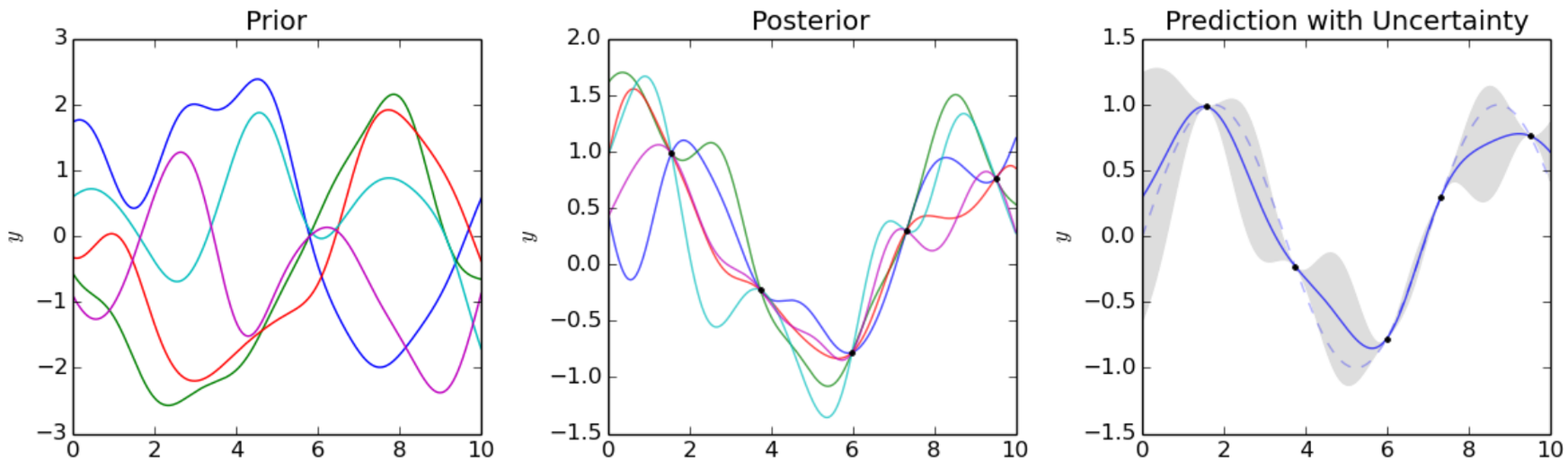
$$\mathbb{P}\begin{pmatrix}\mathbf{y}\\ y_*\end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**}\end{bmatrix}\right)$$

$$\mathbf{K}_{ij} = k(x_i, x_j)$$
$$\mathbf{K}_* = (\ k(x_1, x_*), \cdots, k(x_N, x_*)\ )^T$$
$$\mathbf{K}_{**} = k(x_*, x_*)$$

posterior: $\mathbb{P}(y_*|\mathbf{y}) \sim \mathcal{N}(\mathbf{K}_*^\top \mathbf{K}^{-1}\mathbf{y}\ ,\ \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1}\mathbf{K}_*)$

Rasmussen and Williams, "Gaussian processes for machine learning," MIT Press, 2006.

Gaussian processes



mean — covariance (kernel)

prior: $f(x) \sim \mathcal{GP}(\mu(x),\ k(x, x'))$

$$\mathbb{P}\begin{pmatrix}\mathbf{y}\\ y_*\end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**}\end{bmatrix}\right)$$

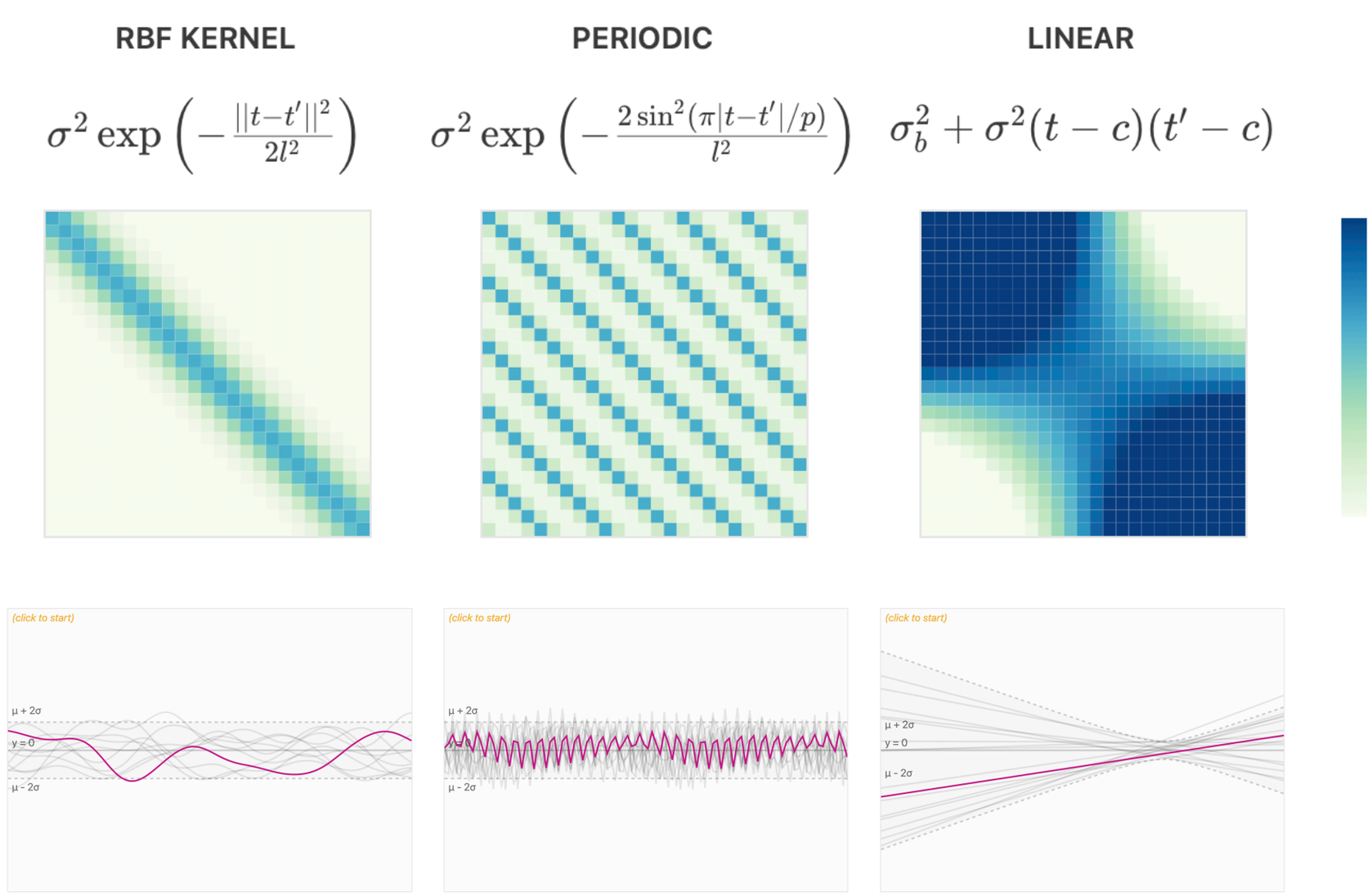
$$\mathbf{K}_{ij} = k(x_i, x_j)$$
$$\mathbf{K}_* = (\ k(x_1, x_*), \cdots, k(x_N, x_*)\ )^T$$
$$\mathbf{K}_{**} = k(x_*, x_*)$$

posterior: $\mathbb{P}(y_*|\mathbf{y}) \sim \mathcal{N}(\mathbf{K}_*^\top \mathbf{K}^{-1}\mathbf{y}\ ,\ \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1}\mathbf{K}_*)$

Rasmussen and Williams, "Gaussian processes for machine learning," MIT Press, 2006.

Gaussian processes

- Example kernels

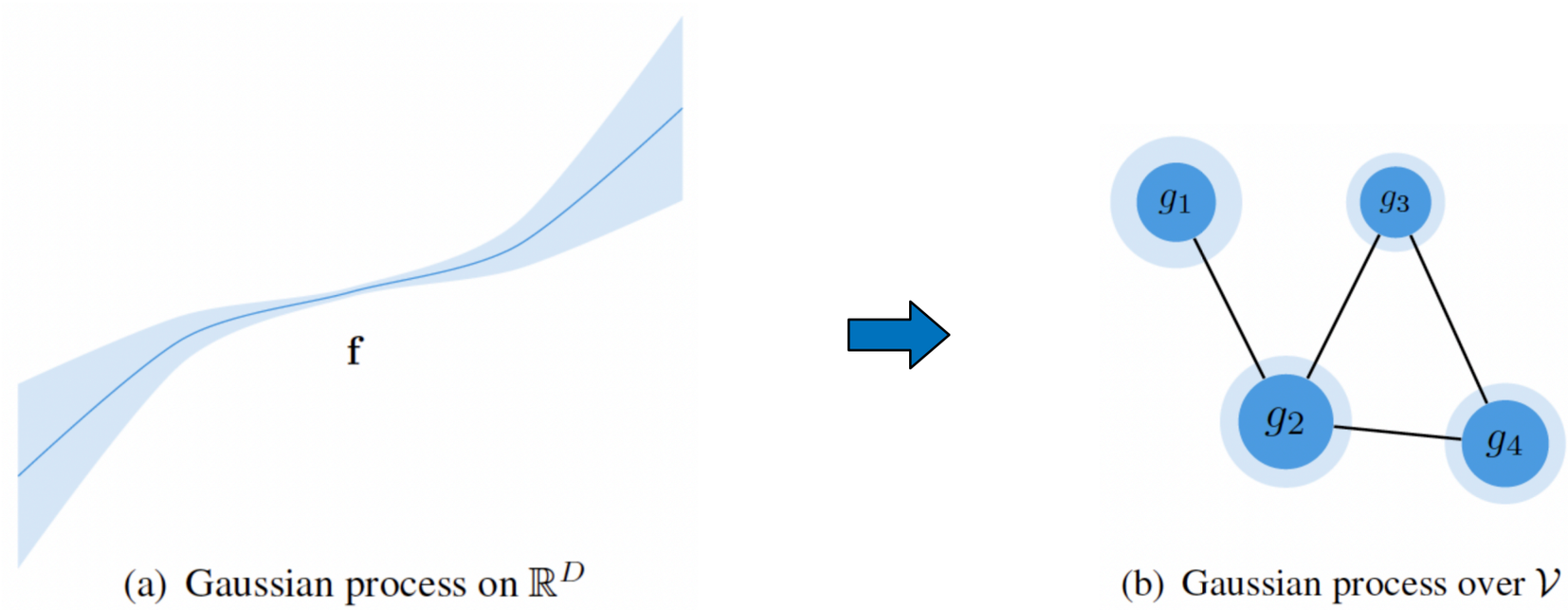


https://distill.pub/2019/visual-exploration-gaussian-processes/

GPs on graphs?

(a) Gaussian process on $\mathbb{R}^D$

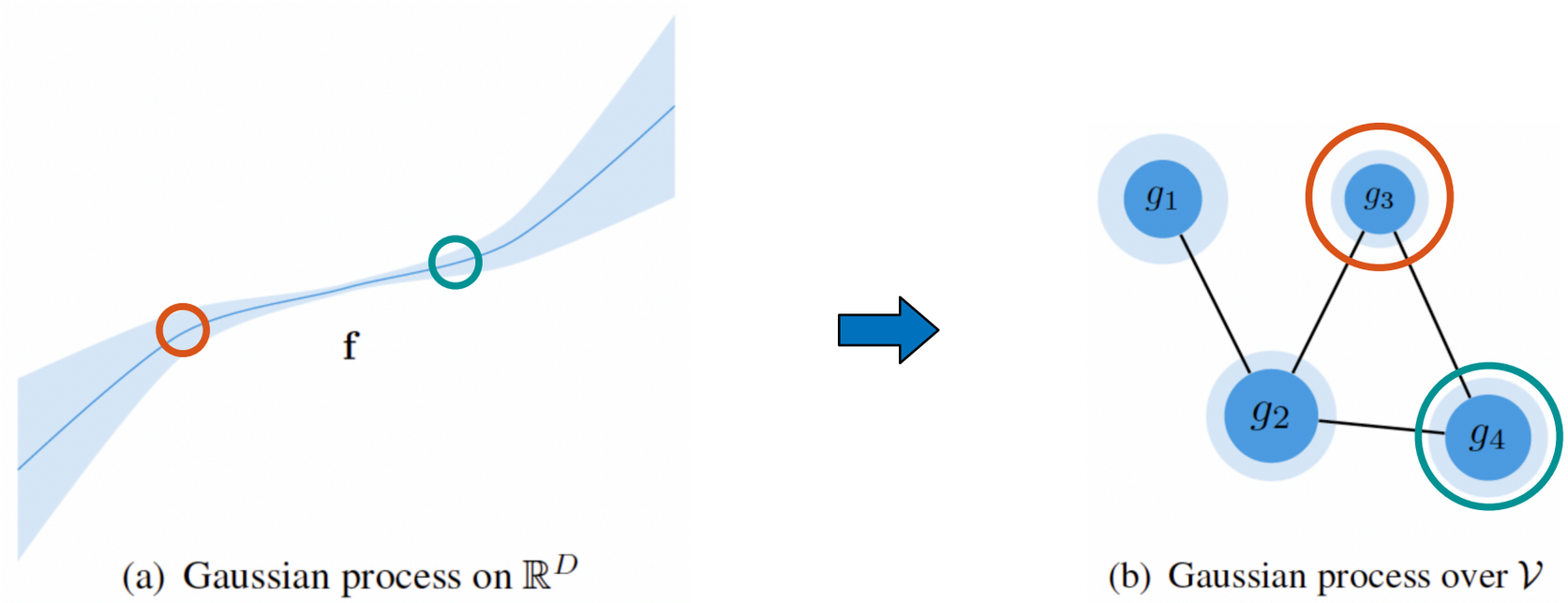
GPs on graphs?



(a) Gaussian process on $\mathbb{R}^D$

(b) Gaussian process over $\mathcal{V}$

GPs on graphs?

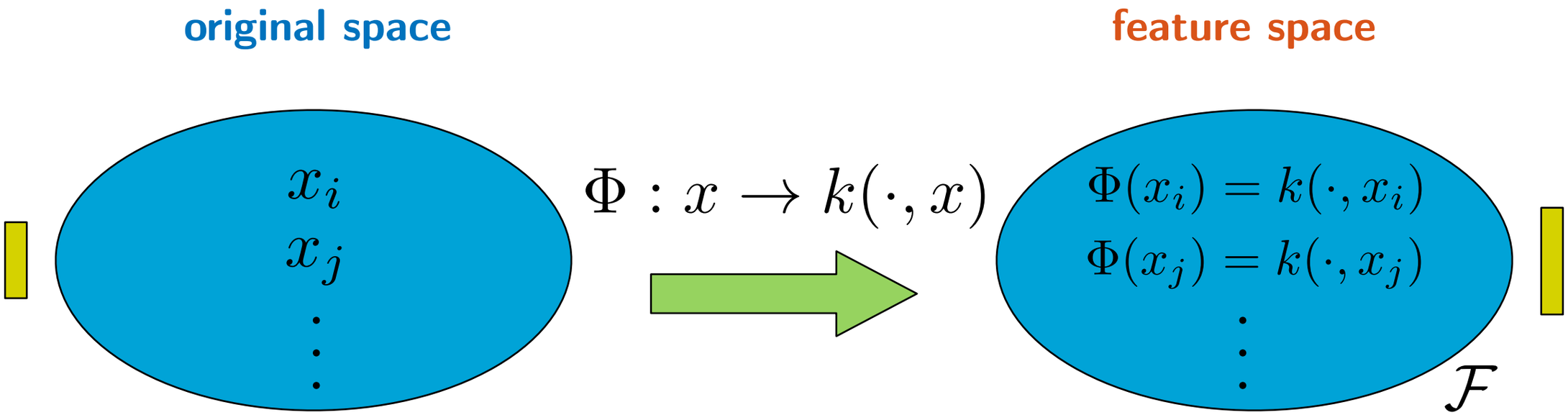


(a) Gaussian process on $\mathbb{R}^D$

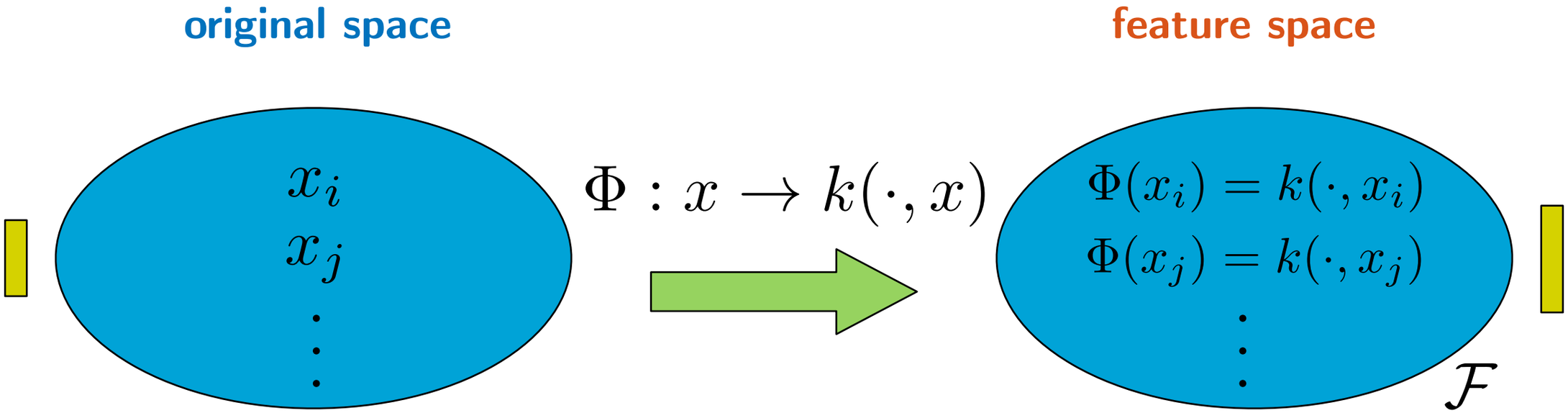
(b) Gaussian process over $\mathcal{V}$

requires definition of kernels on graphs

Reproducing Kernel Hilbert Space



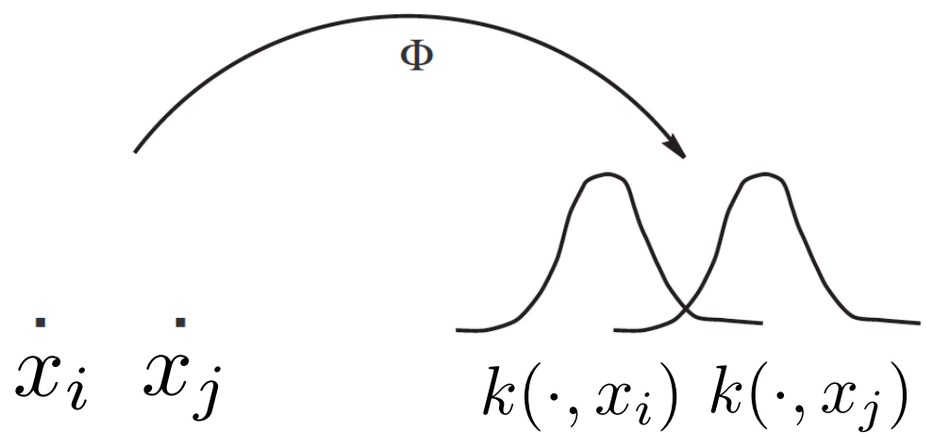
Reproducing Kernel Hilbert Space



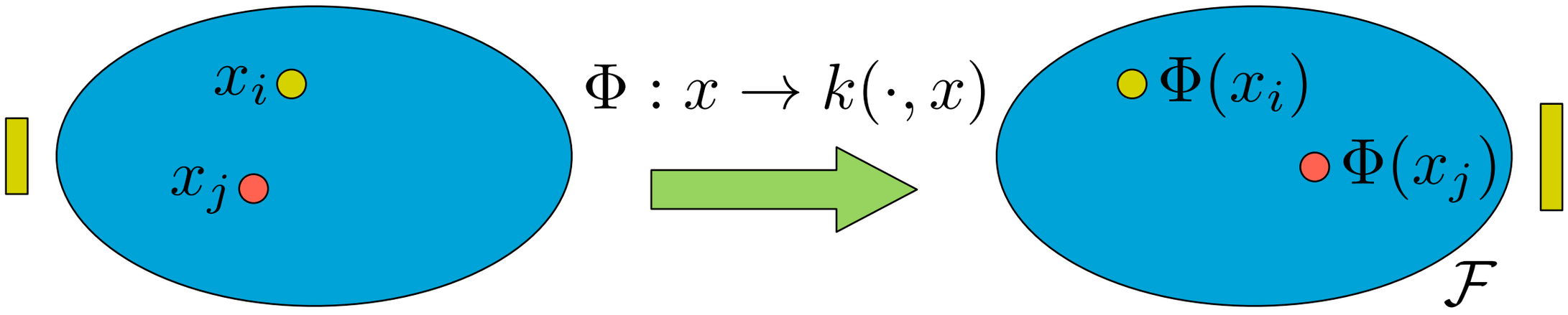
example

$$\Phi(x_i) = k(\cdot, x_i) = exp(-\frac{||\cdot - x_i||^2}{2\sigma^2})$$

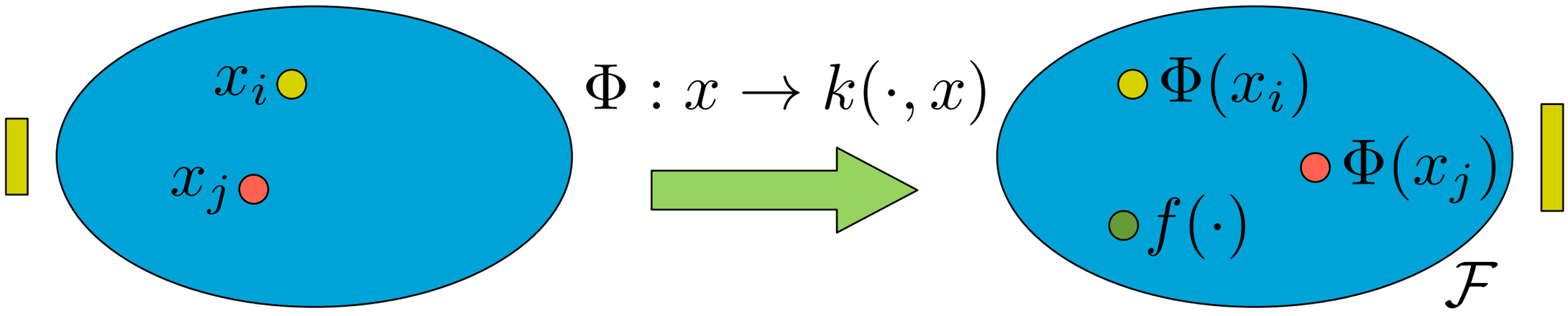
$$\Phi(x_j) = k(\cdot, x_j) = exp(-\frac{||\cdot - x_j||^2}{2\sigma^2})$$



Reproducing Kernel Hilbert Space

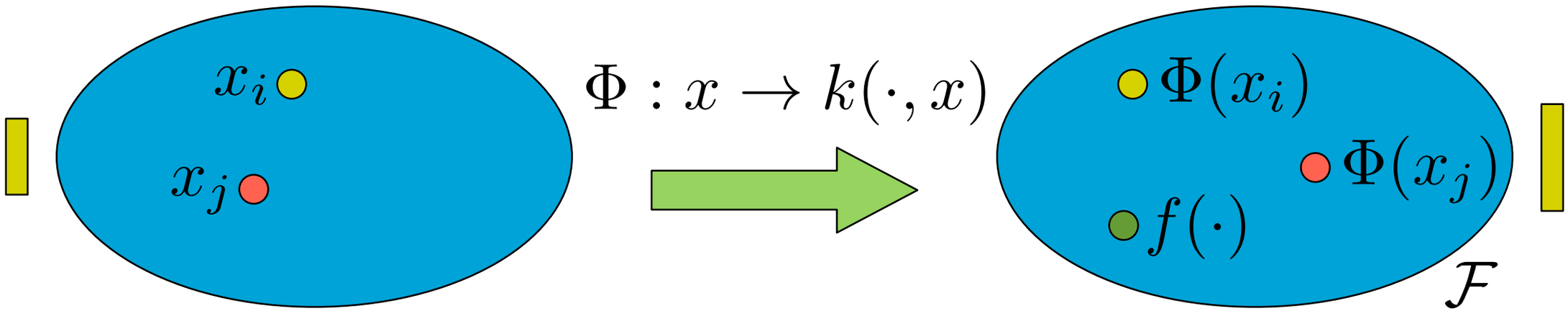


Reproducing Kernel Hilbert Space



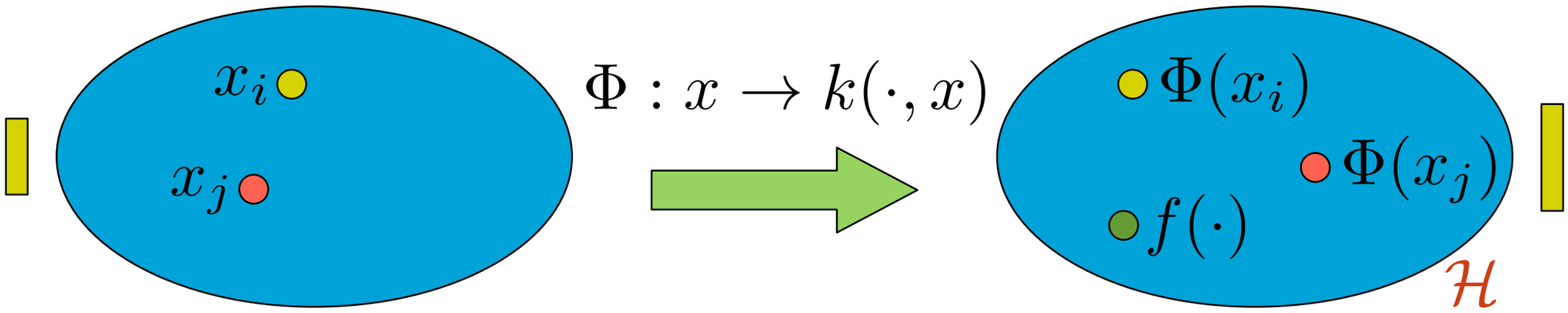
- vector space: $f(\cdot) = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$

Reproducing Kernel Hilbert Space



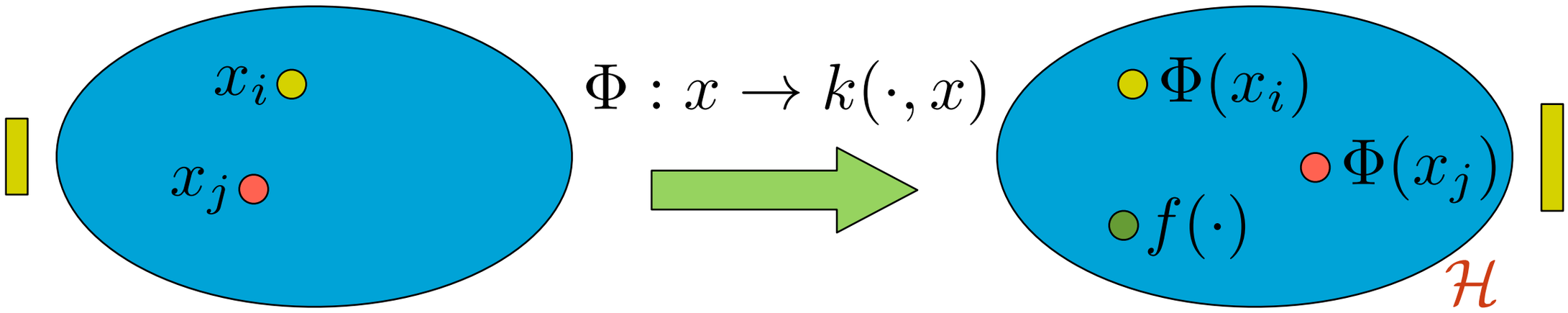
- vector space: $f(\cdot) = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$
- inner product: $\langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{F}} = \langle k(\cdot, x_i), k(\cdot, x_j) \rangle_{\mathcal{F}} := k(x_i, x_j)$

Reproducing Kernel Hilbert Space



- vector space: $f(\cdot) = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$
- inner product: $\langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{F}} = \langle k(\cdot, x_i), k(\cdot, x_j) \rangle_{\mathcal{F}} := k(x_i, x_j)$
- make it complete to obtain an RKHS

Reproducing Kernel Hilbert Space

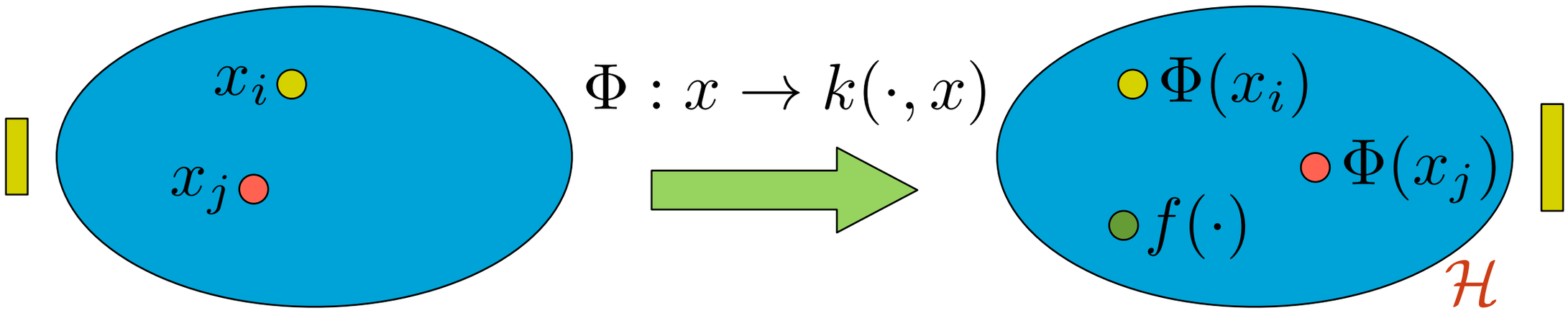


- vector space: $f(\cdot) = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$
- inner product: $\langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{F}} = \langle k(\cdot, x_i), k(\cdot, x_j) \rangle_{\mathcal{F}} := k(x_i, x_j)$
- make it complete to obtain an RKHS

reproducing property

$$\langle f(\cdot), \Phi(x) \rangle_{\mathcal{H}} = \langle \sum_{i=1}^{m} \alpha_i k(\cdot, x_i), k(\cdot, x) \rangle_{\mathcal{H}} = \sum_{i=1}^{m} \alpha_i k(x, x_i) = f(x)$$

Reproducing Kernel Hilbert Space



- vector space: $f(\cdot) = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$
- inner product: $\langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{F}} = \langle k(\cdot, x_i), k(\cdot, x_j) \rangle_{\mathcal{F}} := k(x_i, x_j)$
- make it complete to obtain an RKHS

reproducing property

$$\langle f(\cdot), \Phi(x) \rangle_{\mathcal{H}} = \langle \sum_{i=1}^{m} \alpha_i k(\cdot, x_i), k(\cdot, x) \rangle_{\mathcal{H}} = \sum_{i=1}^{m} \alpha_i k(x, x_i) = f(x)$$

$$\int f(t)\delta(x,t)dt = f(x)$$

Kernels on graphs

Laplace operator: Δ

$$\Omega(\|f\|^2) = \langle Pf, Pf \rangle := \langle f, r(\Delta) f \rangle = \langle f, f \rangle_{\mathcal{H}}$$

$$\langle f(\cdot), k(\cdot, x) \rangle_{\mathcal{H}} = \langle f(\cdot), r(\Delta) k(\cdot, x) \rangle = f(x)$$

$$K = r^{-1}(\Delta)$$

Smola and Kondor, "Kernels and regularization on graphs," COLT, 2003.

Kernels on graphs

Laplace operator: Δ

$$\Omega(\|f\|^2) = \langle Pf, Pf \rangle := \langle f, r(\Delta) f \rangle = \langle f, f \rangle_{\mathcal{H}}$$

$$\langle f(\cdot), k(\cdot, x) \rangle_{\mathcal{H}} = \langle f(\cdot), r(\Delta) k(\cdot, x) \rangle = f(x)$$

$$K = r^{-1}(\Delta)$$

examples:

$k(x, x')$	$r(\Delta)$
$\exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$	$\sum_{i=0}^{\infty} \frac{\sigma^{2i}}{i!} \Delta^i$
$\exp\left(-\frac{1}{\sigma}\|x - x'\|\right)$	$1 + \sigma^2 \Delta$

Smola and Kondor, "Kernels and regularization on graphs," COLT, 2003.

Kernels on graphs

Laplace operator: Δ

$$\Omega(\|f\|^2) = \langle Pf, Pf \rangle := \langle f, r(\Delta) f \rangle = \langle f, f \rangle_{\mathcal{H}}$$

$$\langle f(\cdot), k(\cdot, x) \rangle_{\mathcal{H}} = \langle f(\cdot), r(\Delta) k(\cdot, x) \rangle = f(x)$$

$$K = r^{-1}(\Delta)$$

graph Laplacian: L

$$\langle Pf, Pf \rangle := \langle f, r(L) f \rangle = \langle f, f \rangle_{\mathcal{H}}$$

$$\langle f(\cdot), k(\cdot, v) \rangle_{\mathcal{H}} = \langle f(\cdot), r(L) k(\cdot, v) \rangle = f(v)$$

$$K = r^{-1}(L)$$

examples:

$k(x, x')$	$r(\Delta)$
$\exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$	$\sum_{i=0}^{\infty} \frac{\sigma^{2i}}{i!} \Delta^i$
$\exp\left(-\frac{1}{\sigma}\|x - x'\|\right)$	$1 + \sigma^2 \Delta$

Smola and Kondor, "Kernels and regularization on graphs," COLT, 2003.

Kernels on graphs

Laplace operator: Δ

$$\Omega(\|f\|^2) = \langle Pf, Pf \rangle := \langle f, r(\Delta) f \rangle = \langle f, f \rangle_{\mathcal{H}}$$

$$\langle f(\cdot), k(\cdot, x) \rangle_{\mathcal{H}} = \langle f(\cdot), r(\Delta) k(\cdot, x) \rangle = f(x)$$

$$K = r^{-1}(\Delta)$$

examples:

$k(x, x')$	$r(\Delta)$
$\exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$	$\sum_{i=0}^{\infty} \frac{\sigma^{2i}}{i!} \Delta^i$
$\exp\left(-\frac{1}{\sigma}\|x - x'\|\right)$	$1 + \sigma^2 \Delta$

graph Laplacian: L

$$\langle Pf, Pf \rangle := \langle f, r(L) f \rangle = \langle f, f \rangle_{\mathcal{H}}$$

$$\langle f(\cdot), k(\cdot, v) \rangle_{\mathcal{H}} = \langle f(\cdot), r(L) k(\cdot, v) \rangle = f(v)$$

$$K = r^{-1}(L)$$

takeaway

- kernel is function of graph Laplacian
- it depends on regularisation r

Smola and Kondor, "Kernels and regularization on graphs," COLT, 2003.

Kernels on graphs

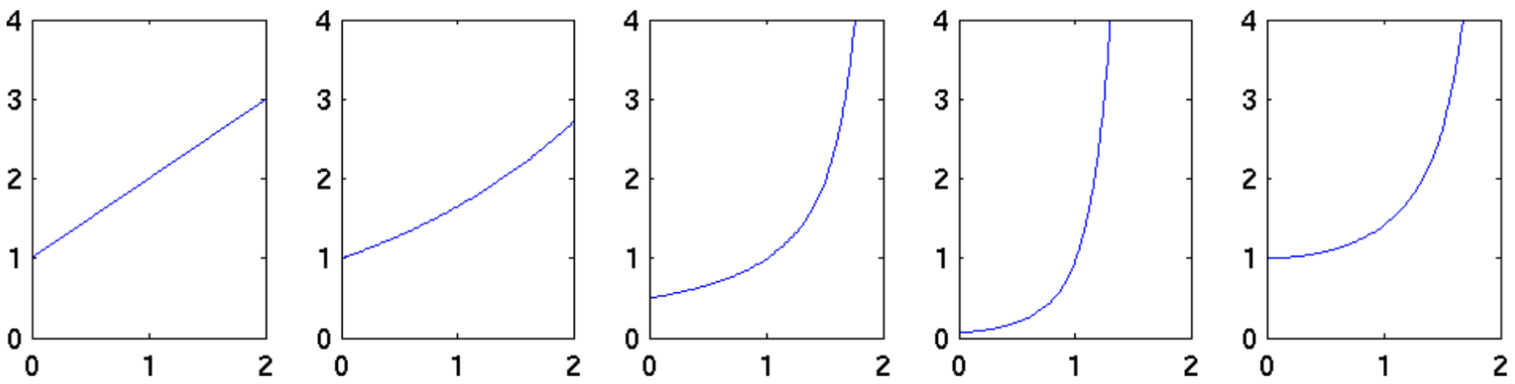


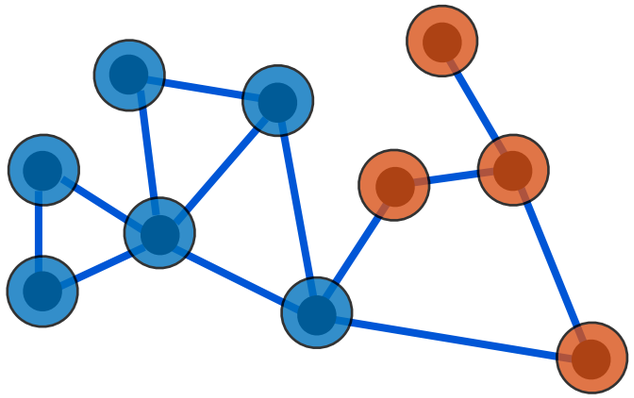
Fig. 1. Regularization function $r(\lambda)$. From left to right: regularized Laplacian ($\sigma^2 = 1$), diffusion process ($\sigma^2 = 1$), one-step random walk ($a = 2$), 4-step random walk ($a = 2$), inverse cosine.

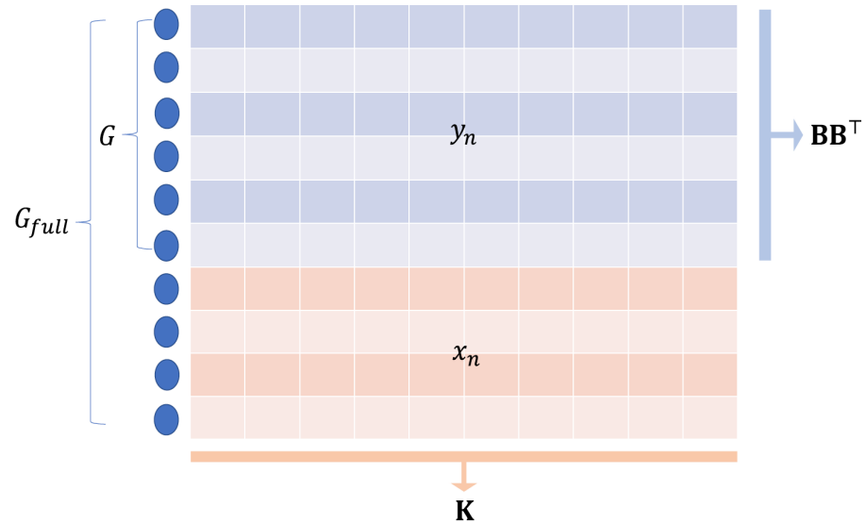
$$
\begin{aligned}
r(\lambda) &= 1 + \sigma^2\lambda \\
r(\lambda) &= \exp\left(\sigma^2/2\lambda\right) \\
r(\lambda) &= (aI - \lambda)^{-p} \text{ with } a \geq 2 \\
r(\lambda) &= (\cos \lambda\pi/4)^{-1}
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
K &= (I + \sigma^2\tilde{L})^{-1} && \text{(Regularized Laplacian)} \\
K &= \exp(-\sigma^2/2\tilde{L}) && \text{(Diffusion Process)} \\
K &= (aI - \tilde{L})^p \text{ with } a \geq 2 && (p\text{-Step Random Walk}) \\
K &= \cos \tilde{L}\pi/4 && \text{(Inverse Cosine)}
\end{aligned}
$$

general form of K: $\sum_{i=1}^{M} r^{-1}(\lambda_i)\mathbf{v}_i\mathbf{v}_i^\top = \mathbf{U}r^{-1}(\mathbf{\Lambda})\mathbf{U}^\top = r^{-1}(\mathbf{L})$

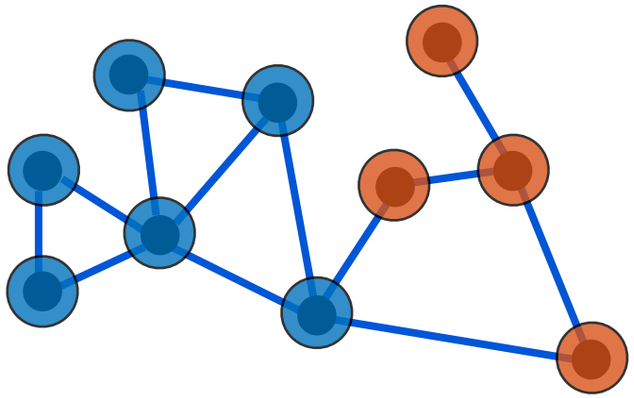
Smola and Kondor, "Kernels and regularization on graphs," COLT, 2003.

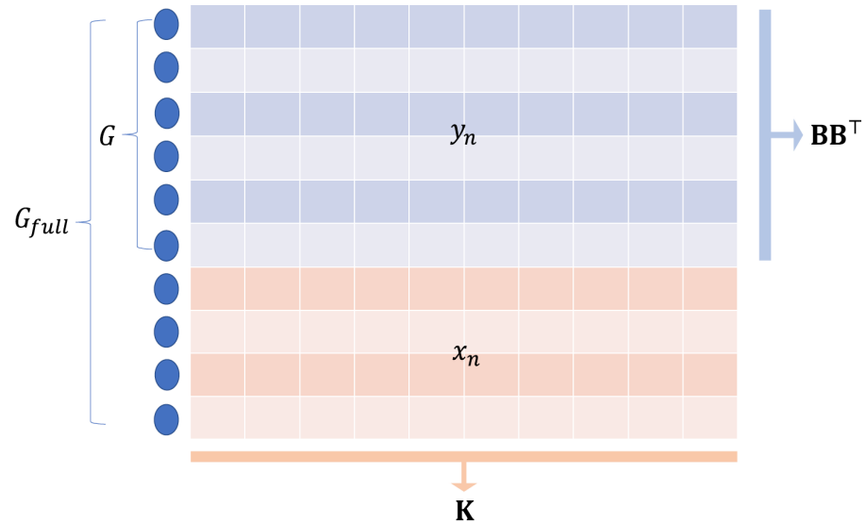
GPs on graphs vis spectra kernel learning





GPs on graphs vis spectra kernel learning





how to combine signal prior and graph information?

GPs on graphs vis spectra kernel learning

signal model: $\mathbf{y}_n = \mathbf{B}\mathbf{f}(\mathbf{x}_n) + \boldsymbol{\epsilon}_n$

Zhi et al., "Gaussian processes on graphs via spectral kernel learning," IEEE TSIPN, 2023.

GPs on graphs vis spectra kernel learning

signal model: $\mathbf{y}_n = \mathbf{B}\mathbf{f}(\mathbf{x}_n) + \boldsymbol{\epsilon}_n$

prior: $\tilde{\mathbf{y}} = \mathrm{vec}([\mathbf{y}_1, \ldots, \mathbf{y}_N]) \quad \mathrm{Cov}(\tilde{\mathbf{y}}) = \mathbf{K} \otimes \mathbf{B}\mathbf{B}^\top + \sigma_\epsilon^2 \mathbf{I}_{MN}$

Zhi et al., "Gaussian processes on graphs via spectral kernel learning," IEEE TSIPN, 2023.

GPs on graphs vis spectra kernel learning

signal model: $$\mathbf{y}_n = \mathbf{B}\mathbf{f}(\mathbf{x}_n) + \boldsymbol{\epsilon}_n$$

prior: $$\tilde{\mathbf{y}} = \mathrm{vec}([\mathbf{y}_1, \ldots, \mathbf{y}_N]) \quad \mathrm{Cov}(\tilde{\mathbf{y}}) = \mathbf{K} \otimes \mathbf{B}\mathbf{B}^\top + \sigma_\epsilon^2 \mathbf{I}_{MN}$$

posterior: $$\mathbb{P}\left(\begin{bmatrix} \tilde{\mathbf{y}} \\ \mathbf{y}_* \end{bmatrix} \right) \sim \mathcal{N}\left(0 \, , \begin{bmatrix} \mathbf{K} \otimes \mathbf{B}\mathbf{B}^\top & \mathbf{K}_* \otimes \mathbf{B}\mathbf{B}^\top \\ \mathbf{K}_*^\top \otimes \mathbf{B}\mathbf{B}^\top & \mathbf{K}_{**} \otimes \mathbf{B}\mathbf{B}^\top \end{bmatrix} + \sigma_\epsilon^2 \mathbf{I}_{M(N+1)} \right)$$

GPs on graphs vis spectra kernel learning

signal model: $$\mathbf{y}_n = \mathbf{B}\mathbf{f}(\mathbf{x}_n) + \boldsymbol{\epsilon}_n$$

prior: $$\tilde{\mathbf{y}} = \mathrm{vec}([\mathbf{y}_1, \ldots, \mathbf{y}_N]) \quad \mathrm{Cov}(\tilde{\mathbf{y}}) = \mathbf{K} \otimes \mathbf{B}\mathbf{B}^\top + \sigma_\epsilon^2 \mathbf{I}_{MN}$$

posterior: $$\mathbb{P}\left(\begin{bmatrix} \tilde{\mathbf{y}} \\ \mathbf{y}_* \end{bmatrix} \right) \sim \mathcal{N}\left(0 \,, \begin{bmatrix} \mathbf{K} \otimes \mathbf{B}\mathbf{B}^\top & \mathbf{K}_* \otimes \mathbf{B}\mathbf{B}^\top \\ \mathbf{K}_*^\top \otimes \mathbf{B}\mathbf{B}^\top & \mathbf{K}_{**} \otimes \mathbf{B}\mathbf{B}^\top \end{bmatrix} + \sigma_\epsilon^2 \mathbf{I}_{M(N+1)} \right)$$

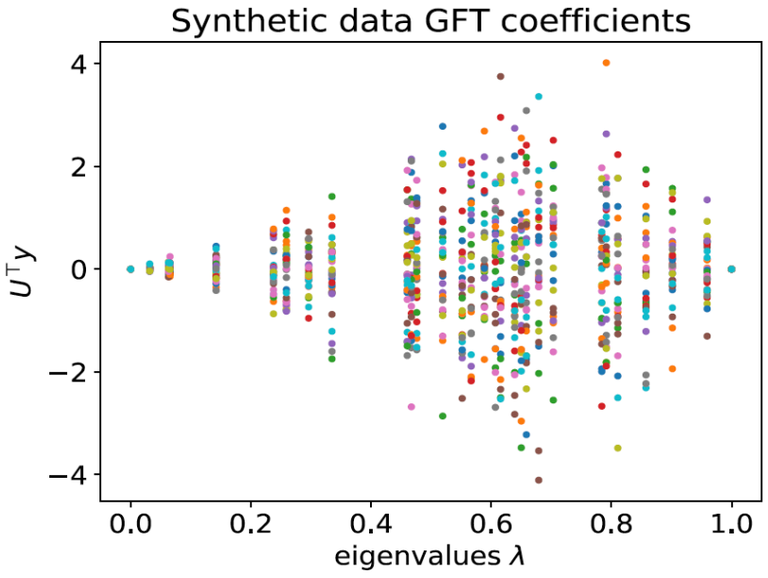
adaptive spectral kernel:

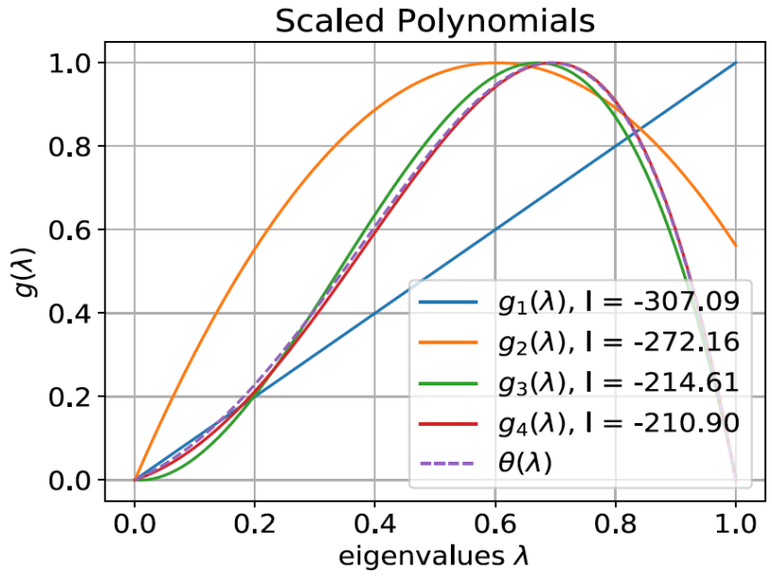
$$\mathbf{B} = \sum_{i=1}^{M} g(\lambda_i)\mathbf{v}_i\mathbf{v}_i^\top = g(\mathbf{L}_S)$$

$$g(\lambda) = \beta_0 + \beta_1\lambda + \cdots + \beta_P\lambda^P \implies \mathbf{B} = \sum_{i=0}^{P} \beta_i \mathbf{L}_S^i$$

Zhi et al., "Gaussian processes on graphs via spectral kernel learning," IEEE TSIPN, 2023.

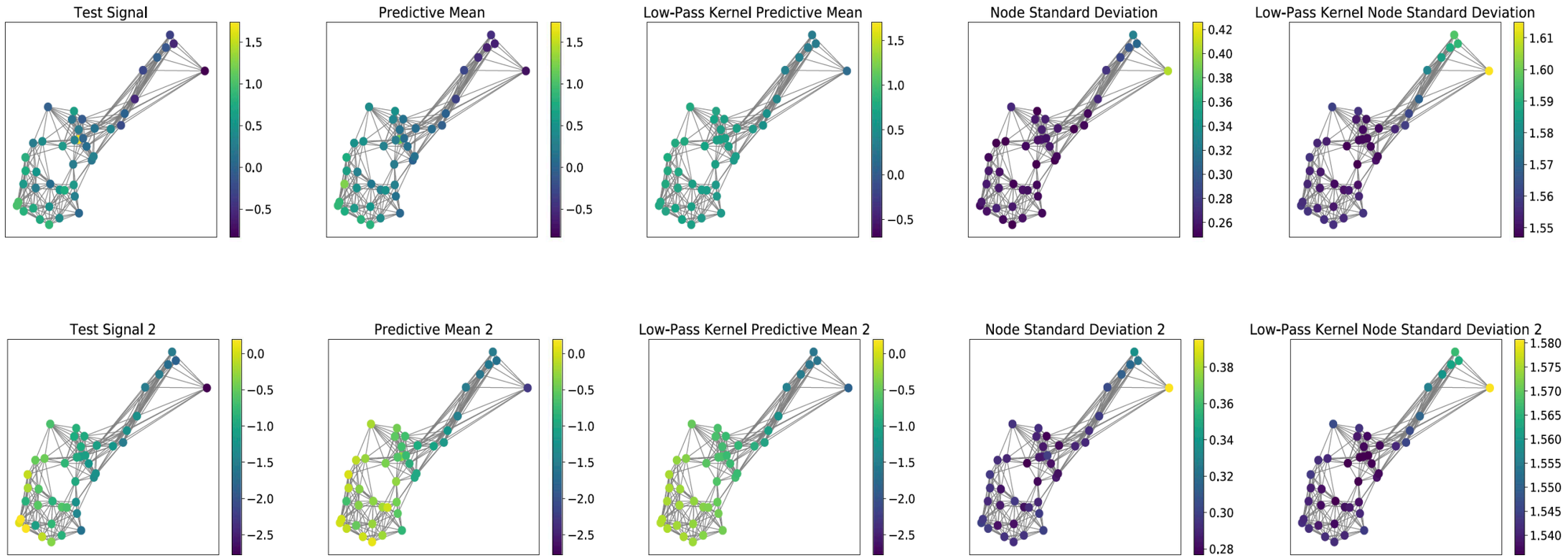
GPs on graphs vis spectra kernel learning





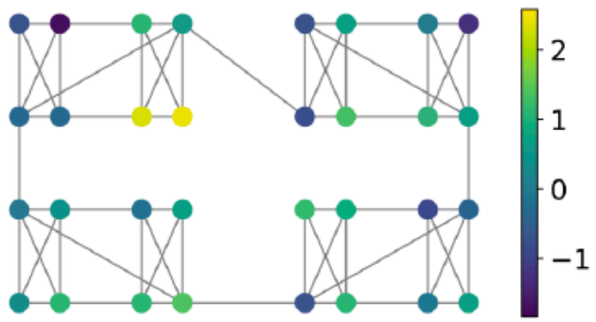
Zhi et al., "Gaussian processes on graphs via spectral kernel learning," IEEE TSIPN, 2023.

GPs on graphs vis spectra kernel learning

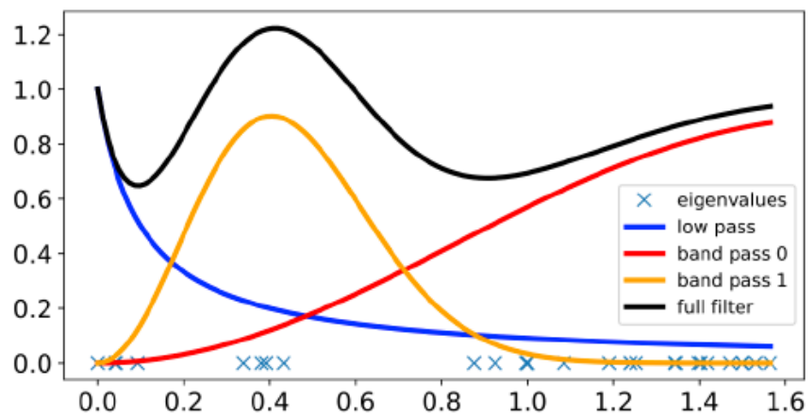


Zhi et al., "Gaussian processes on graphs via spectral kernel learning," IEEE TSIPN, 2023.

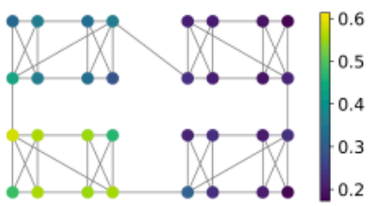
Multi-scale GPs on graphs



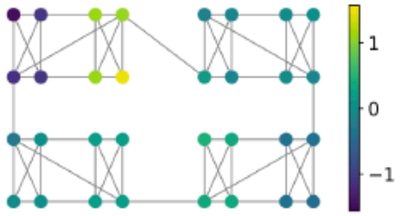
(a) full multi-scale signal



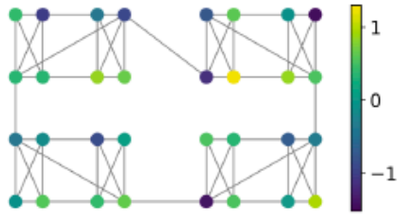
(b) signal spectrum



(c) low-pass filtered signal



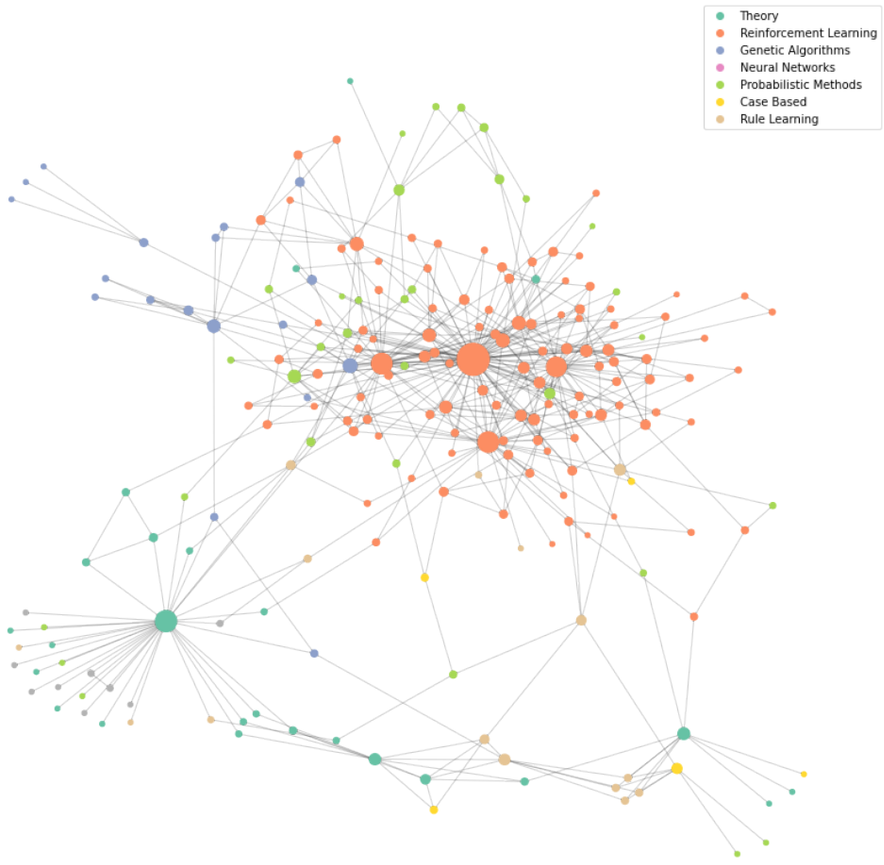
(d) band-pass filtered signal

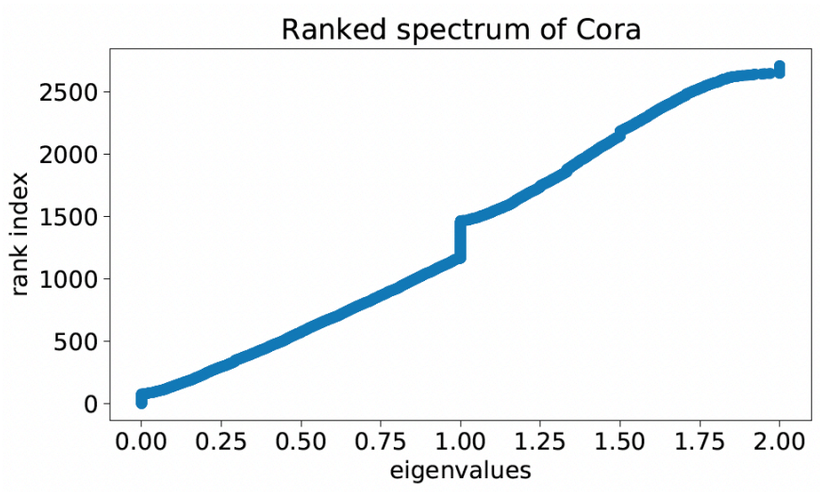


(e) high-pass filtered signal

multi-scale communities and graph signal

Multi-scale GPs on graphs





example: Cora graph and Laplacian spectrum

Multi-scale GPs on graphs

signal model: $\hat{\mathbf{f}} = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{f}$

Multi-scale GPs on graphs

signal model: $$\hat{\mathbf{f}} = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{f}$$

prior: $$\mathbf{W}_\theta := \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \quad \hat{\mathbf{f}} \sim \mathcal{GP}\left(\mathbf{W}_\theta \mathbf{m}, \mathbf{W}_\theta \mathbf{K} \mathbf{W}_\theta^\top\right)$$

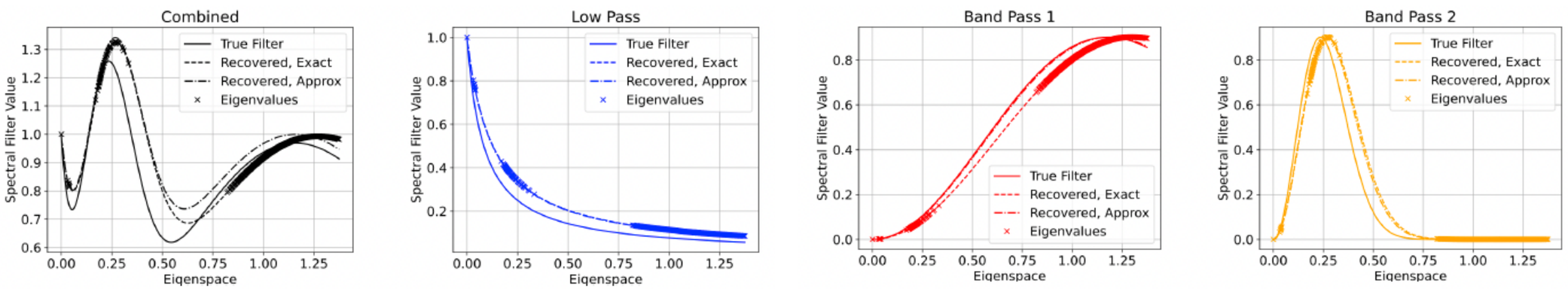
$$g_\theta(\lambda) = h_\alpha(\lambda) + \sum_{l=1}^{L} b_{\beta_l}(\lambda)$$

Multi-scale GPs on graphs

signal model: $\hat{\mathbf{f}} = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{f}$

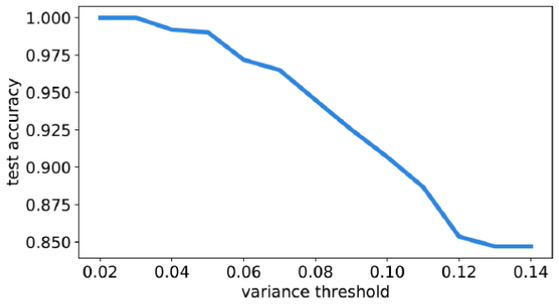
prior: $\mathbf{W}_\theta := \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \quad \hat{\mathbf{f}} \sim \mathcal{GP}\left(\mathbf{W}_\theta \mathbf{m}, \mathbf{W}_\theta \mathbf{K} \mathbf{W}_\theta^\top\right)$

$$g_\theta(\lambda) = h_\alpha(\lambda) + \sum_{l=1}^{L} b_{\beta_l}(\lambda)$$

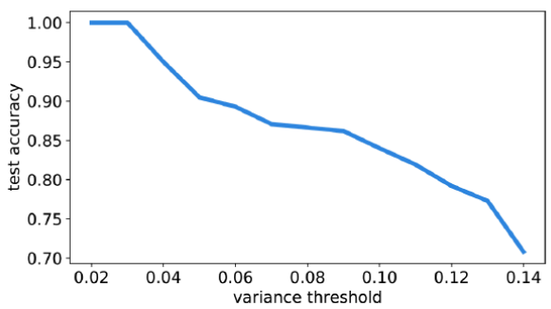


Multi-scale GPs on graphs

Method	Cora	Citeseer	PubMed
GCN (Kipf and Welling, 2017)	80.5 ±0.8	68.1 ±1.3	77.8 ±0.7
GAT (Veličković et al., 2018)	82.6 ±0.7	72.2 ±0.9	76.7 ±0.5
ChebNet (Defferrard et al., 2016)	78.0 ±1.2	70.1 ±0.8	69.8 ±1.1
LanczosNet (Liao et al., 2019)	79.5 ±1.8	66.2 ±1.9	78.3 ±0.3
AdaLanczosNet (Liao et al., 2019)	80.4 ±1.1	68.7 ±1.0	78.1 ±0.4
GP (Ng et al., 2018)	60.8	54.7	71.5
GGP (Ng et al., 2018)	80.9	69.7	77.1
GGP-X (Ng et al., 2018)	84.7	75.6	82.4
ChebGP (ours)	79.7	66.5	77.2
WGGP (ours)	84.7	70.8	78.4
WGGP-X (ours)	87.5	76.8	90.0



(a) Cora



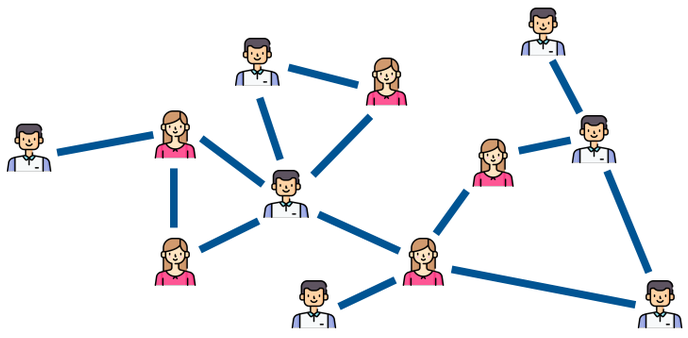
(b) Citeseer

predictive performance and variance (uncertainty)

Lecture 4

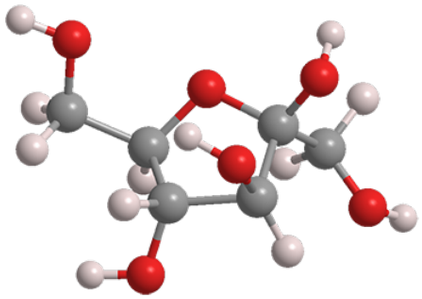
- Gaussian processes on graphs
- Bayesian optimisation of graph-based functions
- Discussion

Graph structures are pervasive

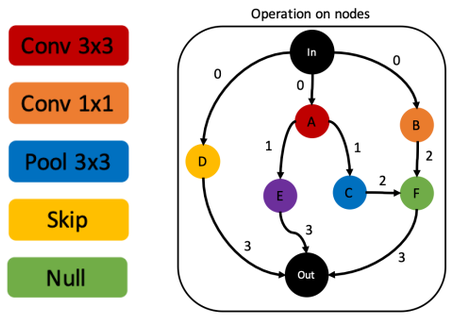


social network

transportation infrastructure



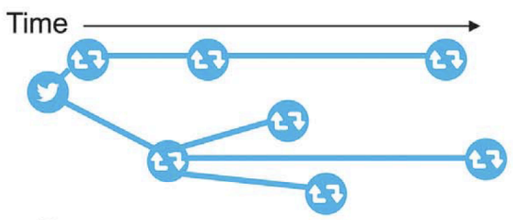
chemical structure



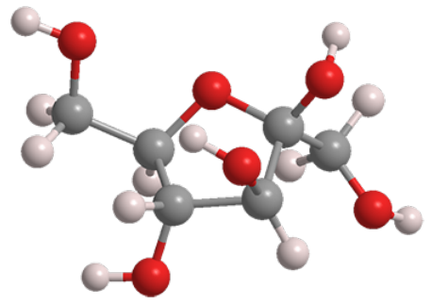
computational architecture

Graph-based functions are pervasive

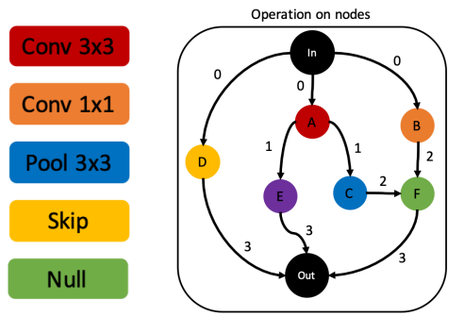
- Graph-wise functions



function: veracity of information in original post
task: misinformation detection



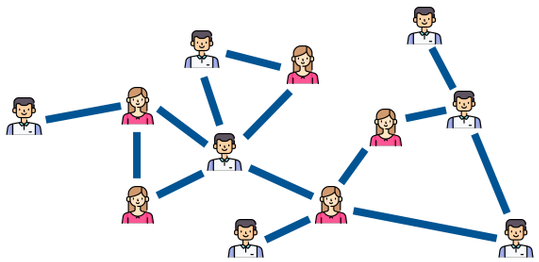
function: potential of molecule inhibiting bacteria
task: molecule discovery



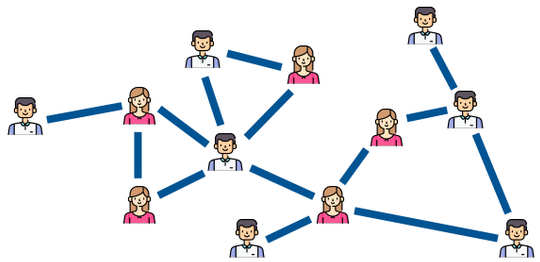
function: test performance of cell-based architecture
task: architecture search

Graph-based functions are pervasive

- Node-wise functions



function: influencing power of individuals
task: influence maximisation



function: infection time of individuals
task: finding "patient zero"

function: criticality of road junctions
task: resilience monitoring

Optimisation of graph-based functions

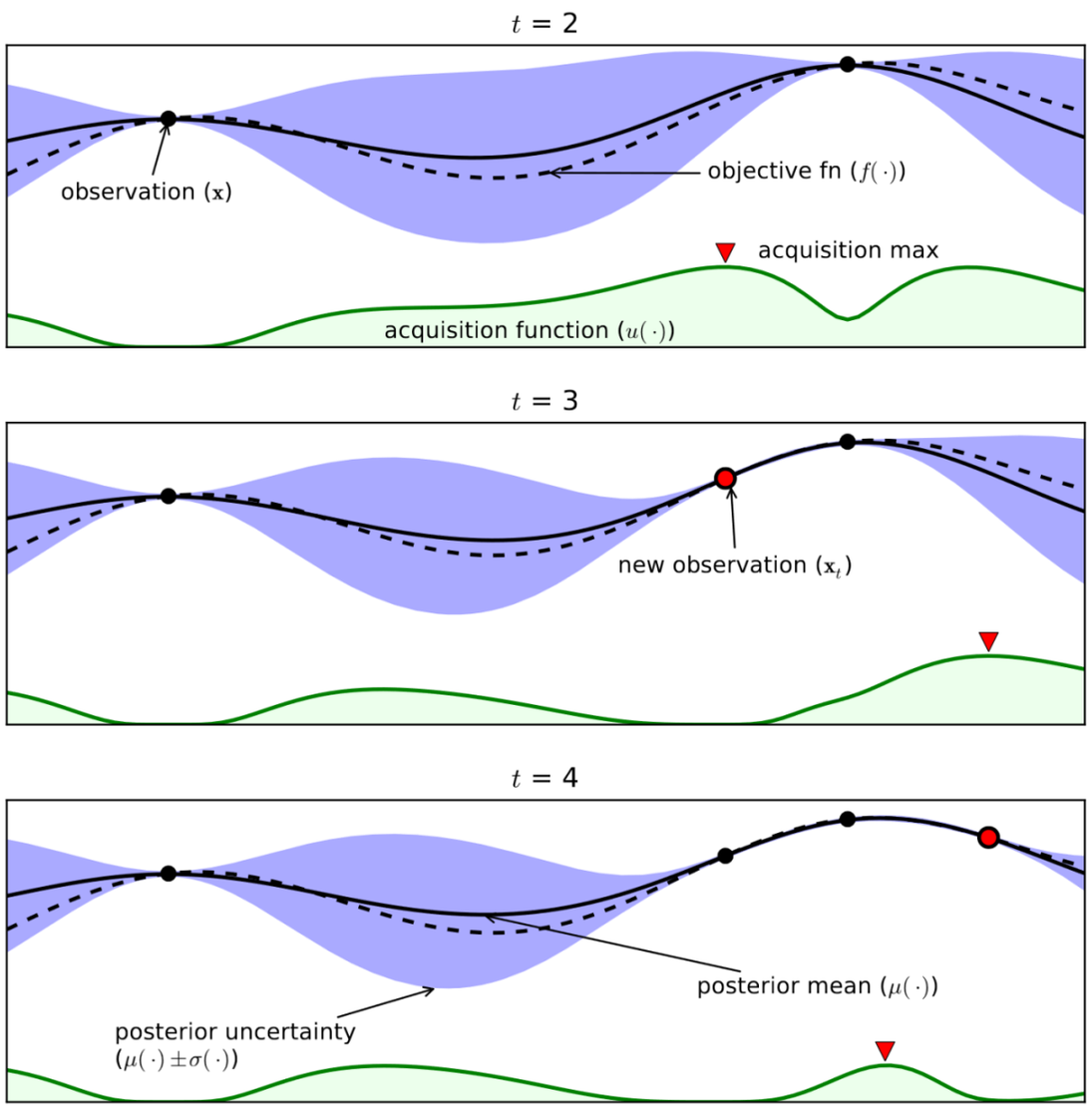
- Diverse problem formulations
  - domain of function can be graphs, nodes or edges
- Search space is challenging
  - discrete and combinatorial search space
  - large number of candidates
  - for node-wise functions, graph may not be directly observable (e.g., epidemiological contact network, offline social network)
- Functions are often **black-box** and **expensive to evaluate**
  - no analytical form or gradient information
  - often requires simulation or real-world intervention

Optimisation of graph-based functions

- Diverse problem formulations
  - domain of function can be graphs, nodes or edges
- Search space is challenging
  - discrete and combinatorial search space
  - large number of candidates
  - for node-wise functions, graph may not be directly observable (e.g., epidemiological contact network, offline social network)
- Functions are often **black-box** and **expensive to evaluate**
  - no analytical form or gradient information
  - often requires simulation or real-world intervention

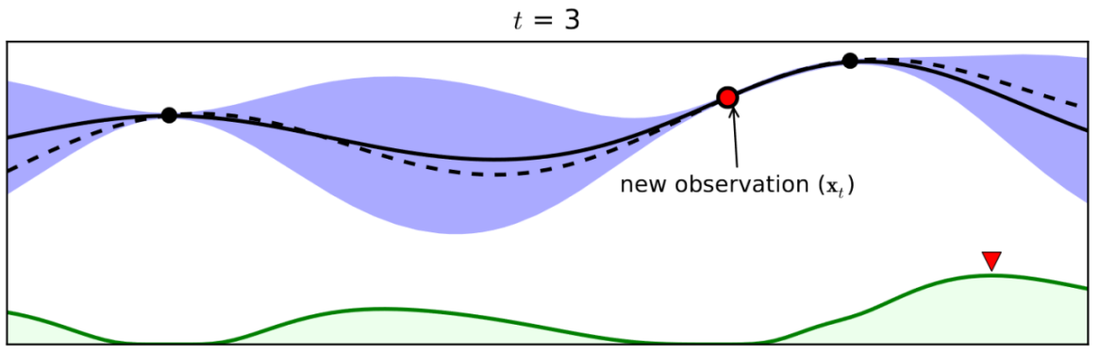
Bayesian optimisation: natural choice, but how to apply in graph case?

Bayesian optimisation

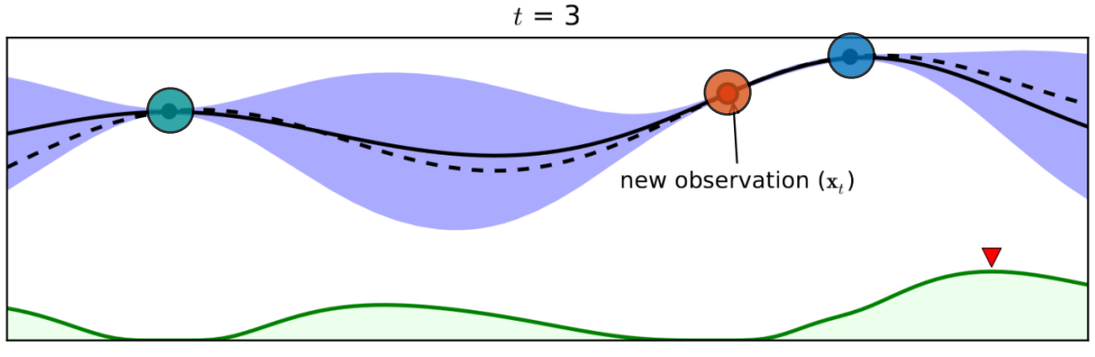


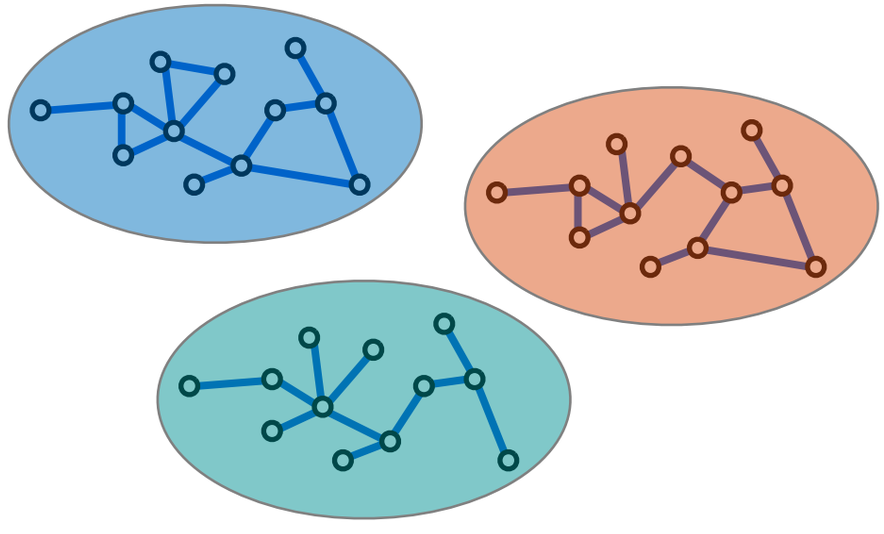
1. query function value at currently chosen point
2. fit **surrogate model**, e.g., Gaussian processes:
   $f(x) \sim \mathcal{GP}(\mu(x), k(x, x'))$
3. use **acquisition function** to find next query point, e.g., upper confidence bound:
   $a_{\mathrm{UCB}}(x; \beta) = u(x) + \beta\sigma(x)$
4. repeat steps 1-3 until query budget exhausts

BO for graph-based functions



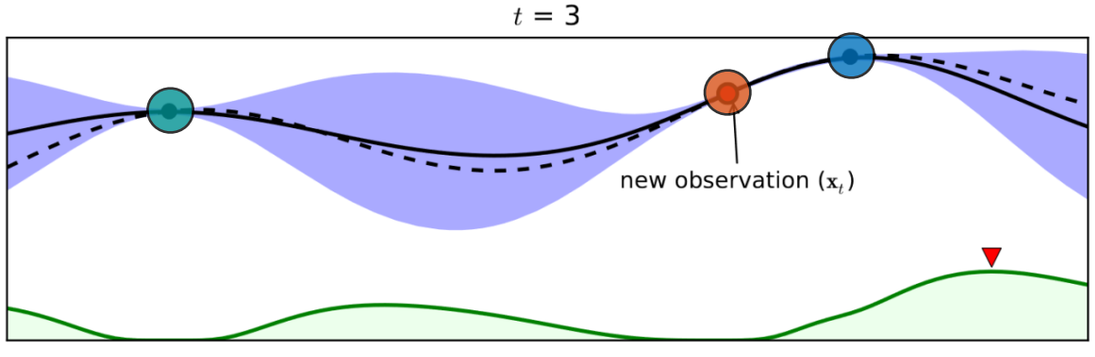
BO for graph-based functions

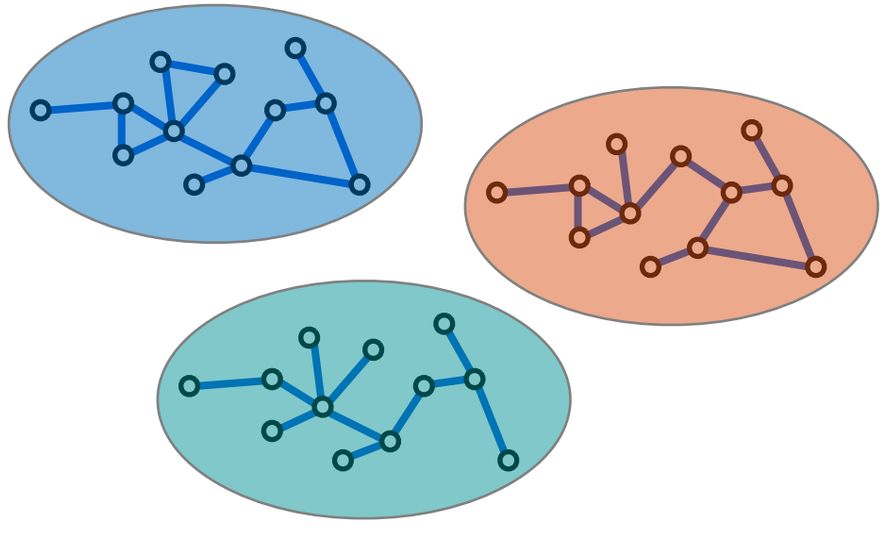




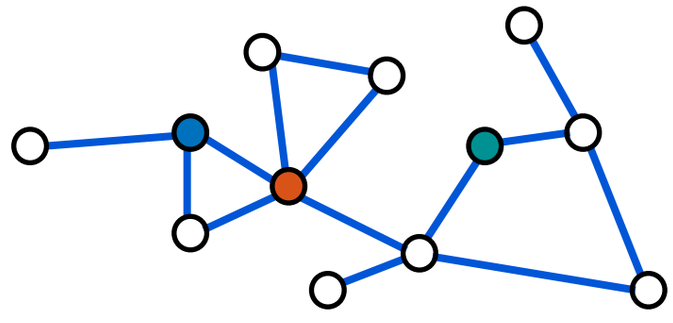
graph-wise function $f(G)$

BO for graph-based functions





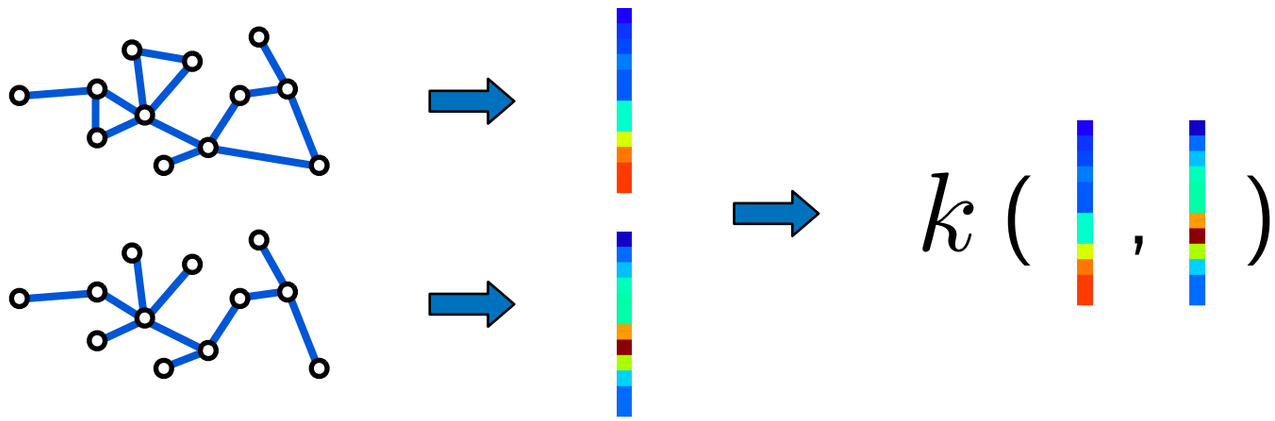
graph-wise function $f(G)$



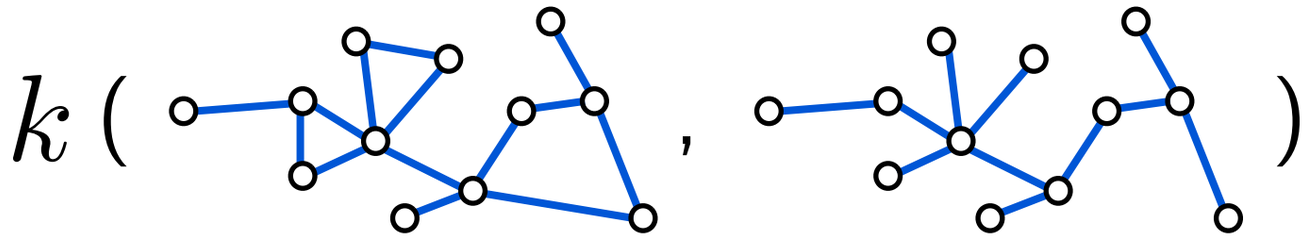
node-wise function $f(v)$

Kernels for graph-based functions

- Graph-wise functions
  - key is comparison **between two graphs**
  - idea 1: embed graphs into vector space and use classical kernels



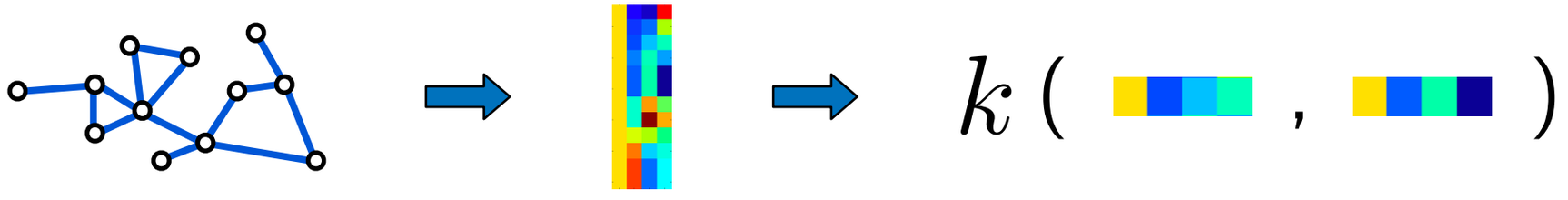
  - idea 2: use graph kernels (e.g., via bag of structures or information propagation)



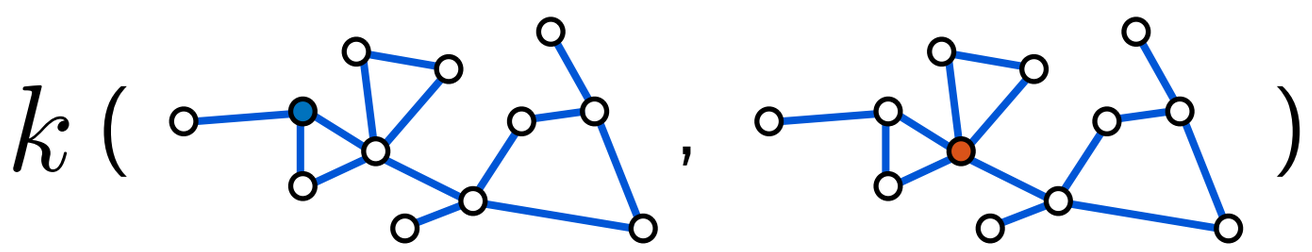
Borgwardt et al., "Graph kernels: State-of-the-art and future challenges," Now Publishers, 2020.

Kernels for graph-based functions

- Node-wise functions
  - key is comparison **between two nodes in the graph**
  - idea 1: embed nodes in the graph into vector space and use classical kernels



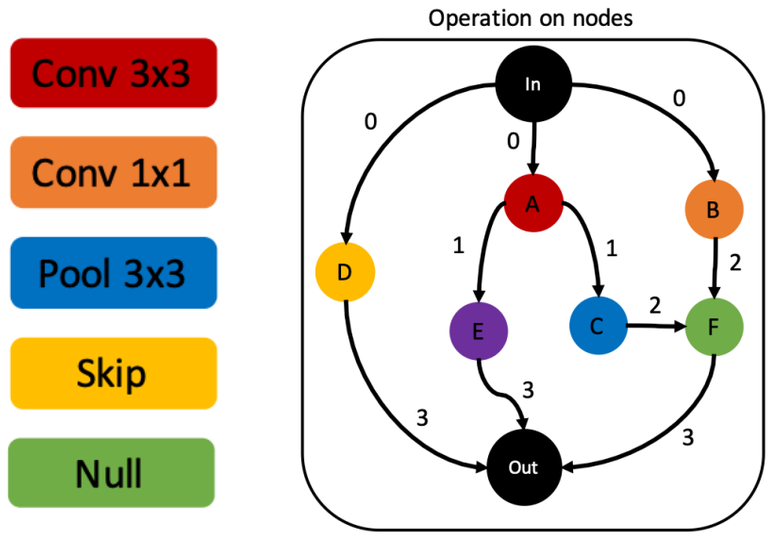
  - idea 2: use kernels defined on graphs (e.g., functions of graph Laplacian)



Smola and Kondor, "Kernels and regularization on graphs," COLT, 2003.

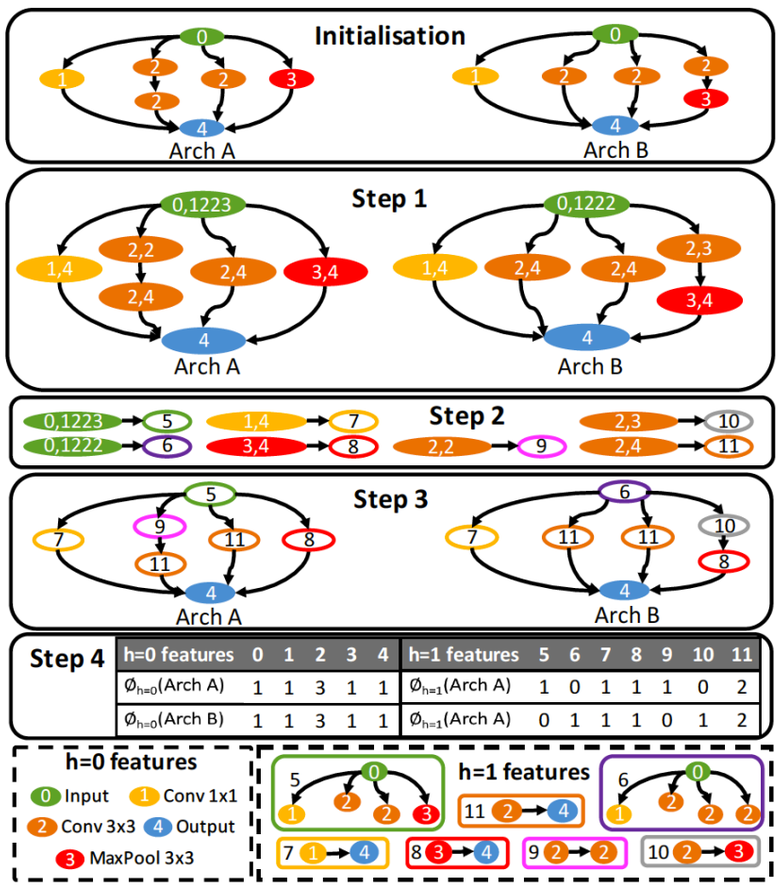
Case 1: Neural architecture search

- Objective: search for effective cell-based neural architecture



$$G^* = \arg\max_{G \in \mathcal{G}} \; f(G)$$

Case 1: Neural architecture search



Methodology

- surrogate model: Gaussian process via Weisfeiler-Lehman graph kernel measuring **similarity between graphs**
- candidate generation: mutation algorithm
- acquisition function: expected improvement

Case 1: Neural architecture search

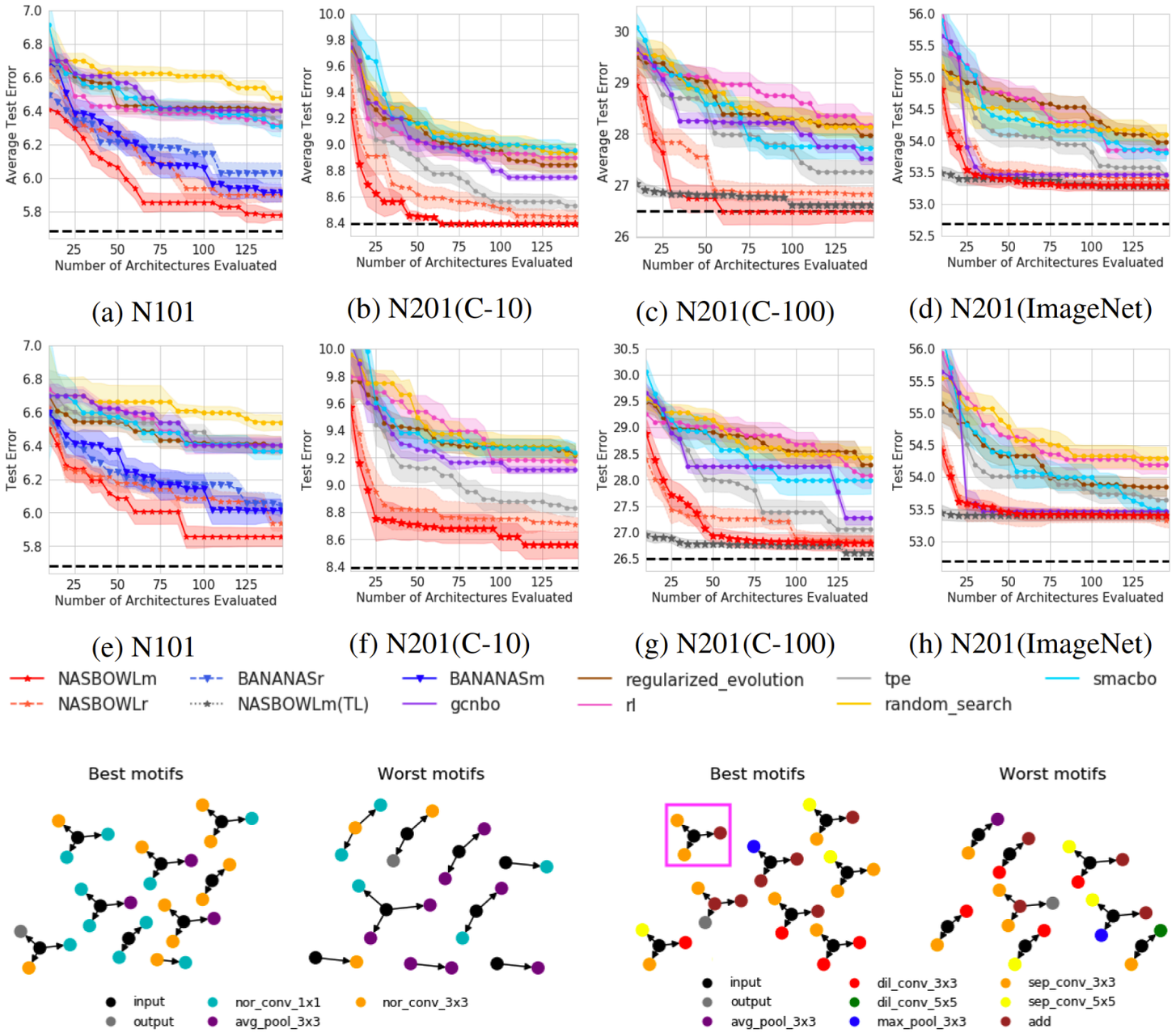
Algorithm 1 NAS-BOWL Algorithm.

1: **Input:** Maximum BO iterations T, BO batch size b, acquisition function $\alpha(\cdot)$, initial observed data on the target task $\mathcal{D}_0$, Optional: past-task query data $\mathcal{D}_{\text{past}}$ and surrogate $\mathcal{S}_{\text{past}}$
2: **Output:** The best architecture G_T^*
3: Initialise the GPWL surrogate $\mathcal{S}$ with $\mathcal{D}_0$
4: **for** $t = 1, \ldots, T$ **do**
11: Generate B candidate architectures $\mathcal{G}_t$
13: $\{G_{t,i}\}_{i=1}^b = \arg\max_{G \in \mathcal{G}_t} \alpha_t(G|\mathcal{D}_{t-1})$
14: Evaluate their validation accuracy $\{y_{t,i}\}_{i=1}^b$
15: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\{G_{t,i}\}_{i=1}^b, \{y_{t,i}\}_{i=1}^b)$
16: Update the surrogate $\mathcal{S}$ with $\mathcal{D}_t$
17: **end for**
18: Return the best architecture seen so far G_T^*

Methodology

- surrogate model: Gaussian process via Weisfeiler-Lehman graph kernel measuring **similarity between graphs**
- candidate generation: mutation algorithm
- acquisition function: expected improvement

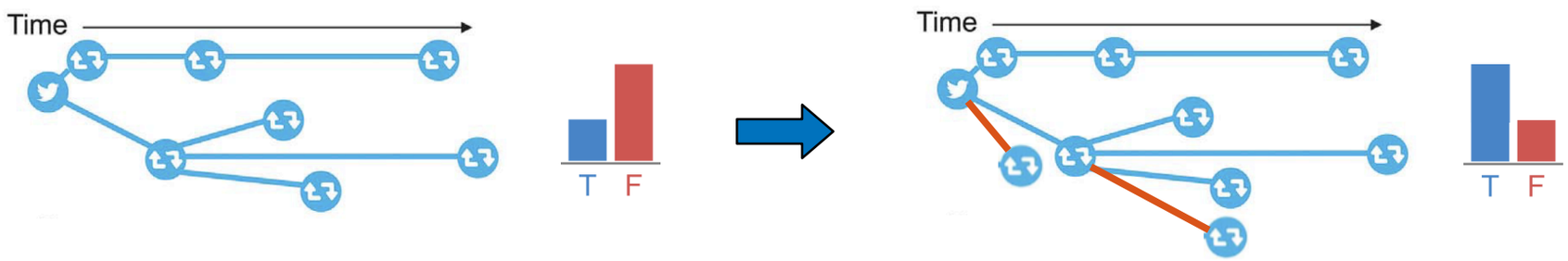
Case 1: Neural architecture search



Ru et al., "Interpretable neural architecture search via Bayesian optimisation with Weisfeiler-Lehman kernels," ICLR, 2021.

Case 2: Adversarial attacks on graphs

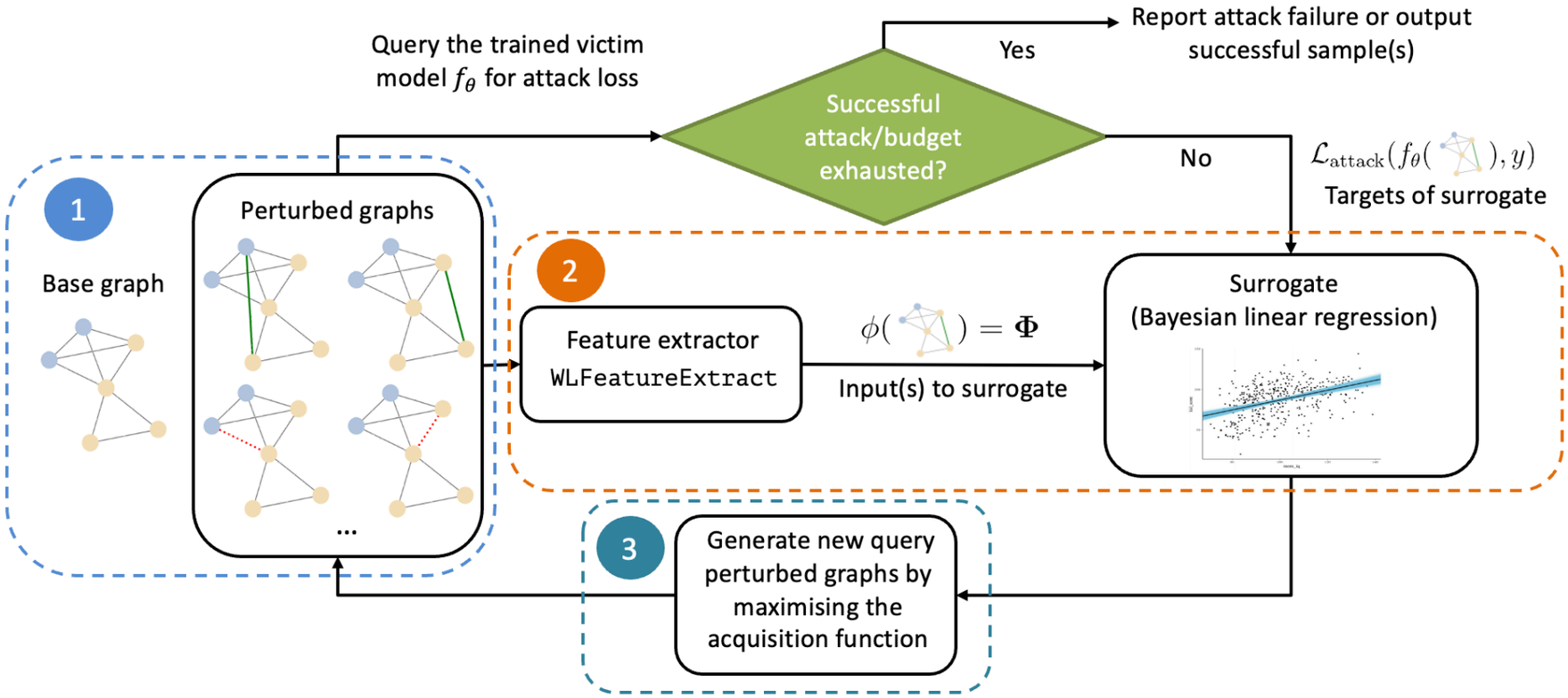
- Objective: find perturbation most harmful to trained graph classifier



$$G^* = \arg\max_{G \in \mathcal{G}} \; \mathcal{L}_{\text{attack}}(f_\theta(G), y)$$

Case 2: Adversarial attacks on graphs

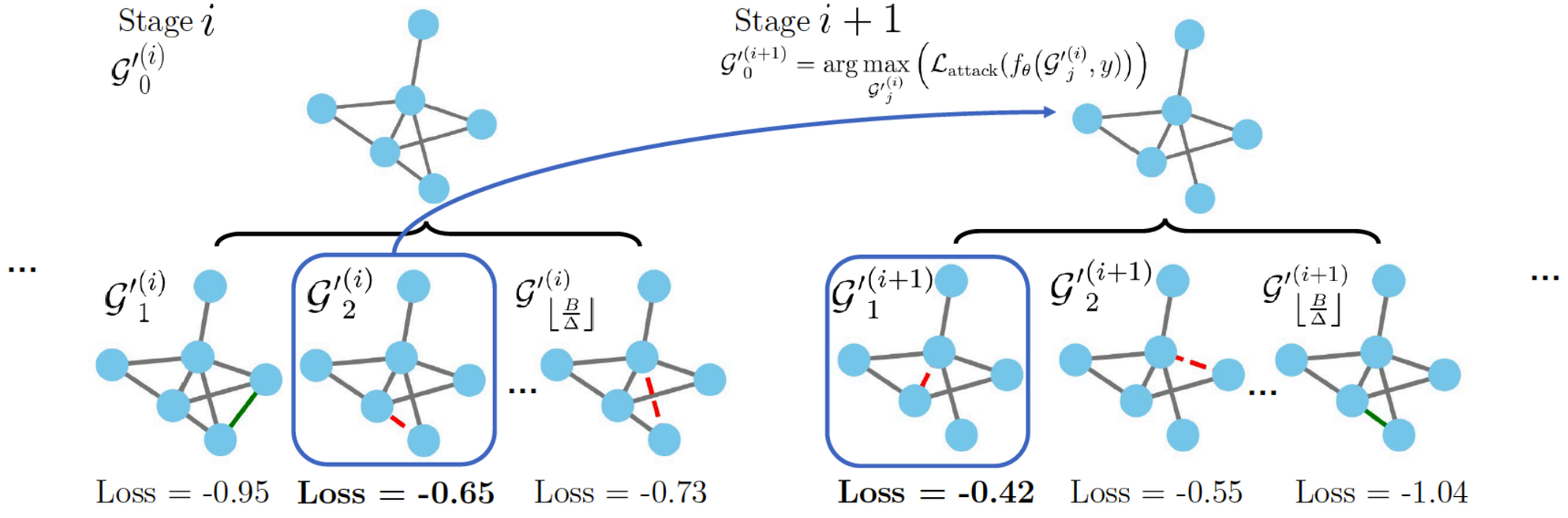
- Methodology
  - surrogate model: Bayesian linear regression with Weisfeiler-Lehman features
  - candidate generation: mutation by edge edit distance 1 from current evaluation
  - acquisition function: expected improvement



Wan et al., "Adversarial attacks on graph classifiers via Bayesian optimisation," NeurIPS, 2021.

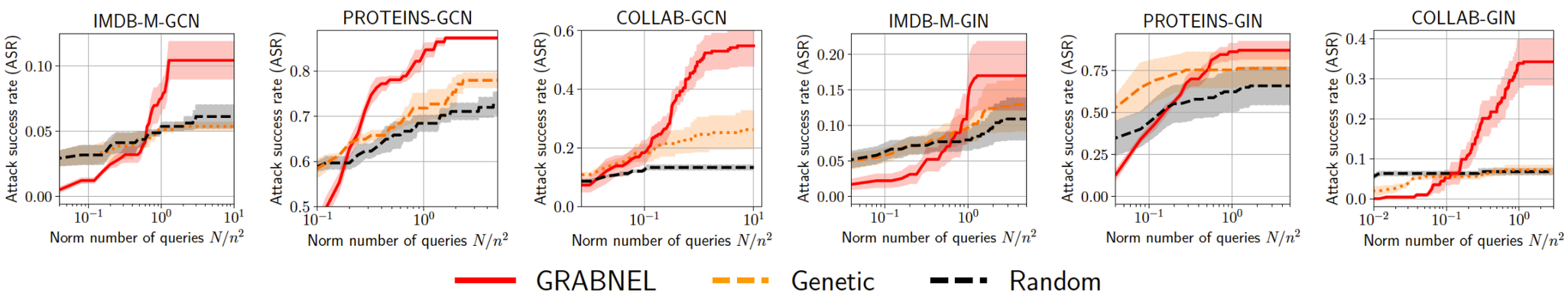
Case 2: Adversarial attacks on graphs

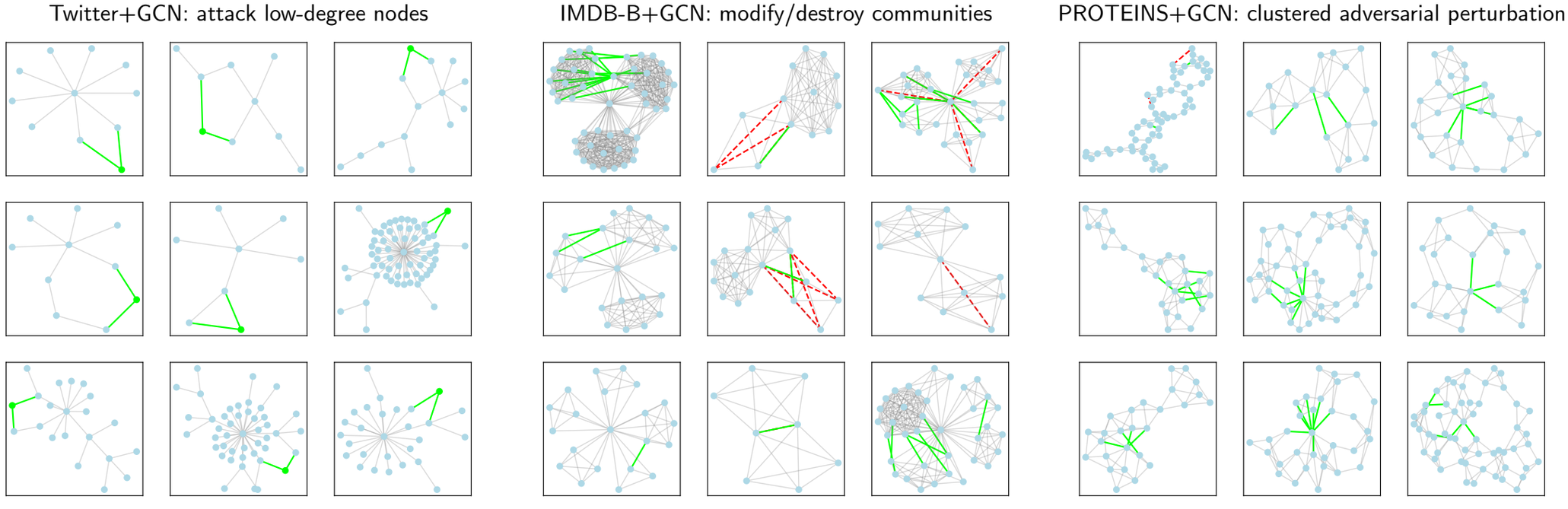
- Methodology
  - surrogate model: Bayesian linear regression with Weisfeiler-Lehman features
  - candidate generation: mutation by edge edit distance 1 from current evaluation
  - acquisition function: expected improvement



Wan et al., "Adversarial attacks on graph classifiers via Bayesian optimisation," NeurIPS, 2021.

Case 2: Adversarial attacks on graphs

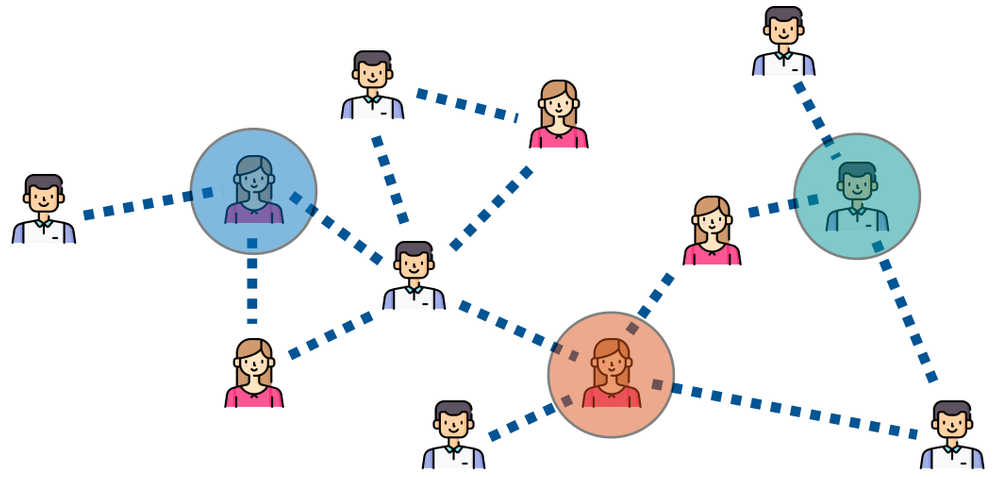




Wan et al., "Adversarial attacks on graph classifiers via Bayesian optimisation," NeurIPS, 2021.

Case 3: Identifying important nodes

- Objective: find node with maximum value of function on node set

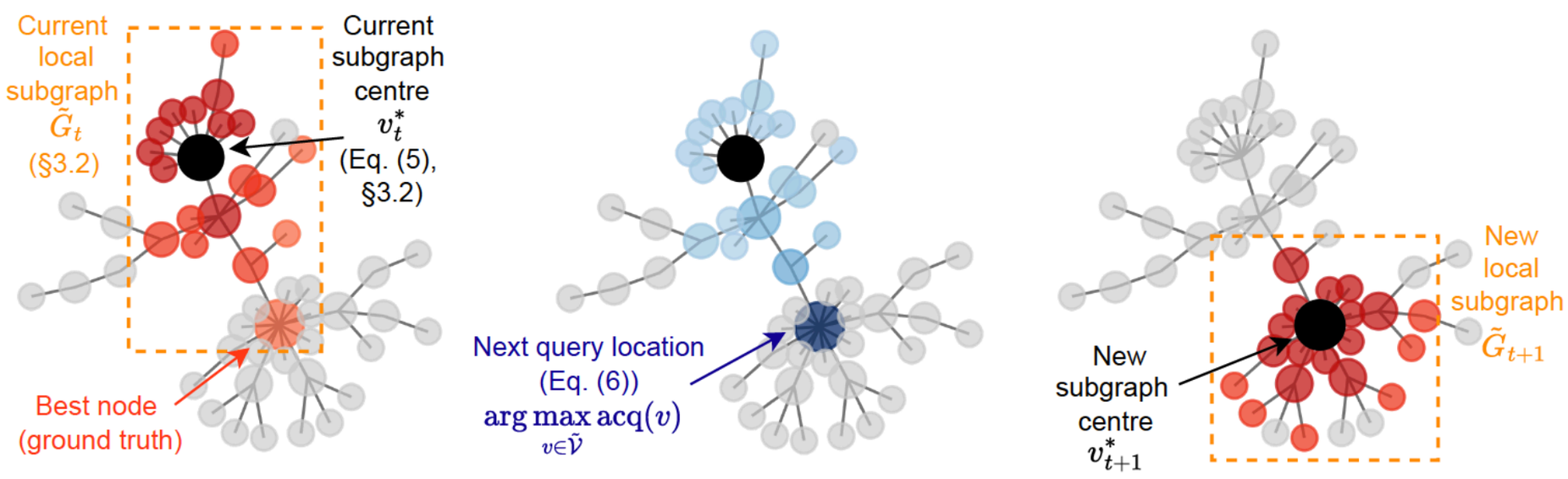


$$v^* = \arg\max_{v \in \mathcal{V}} \; f(v)$$

Wan et al., "Bayesian optimisation of functions on graphs," NeurIPS, 2023.

Case 3: Identifying important nodes

- Methodology
  - progressively sample subgraph on-the-fly
  - surrogate model: Gaussian process via kernels measuring **similarity between nodes**
  - acquisition function: expected improvement



Wan et al., "Bayesian optimisation of functions on graphs," NeurIPS, 2023.

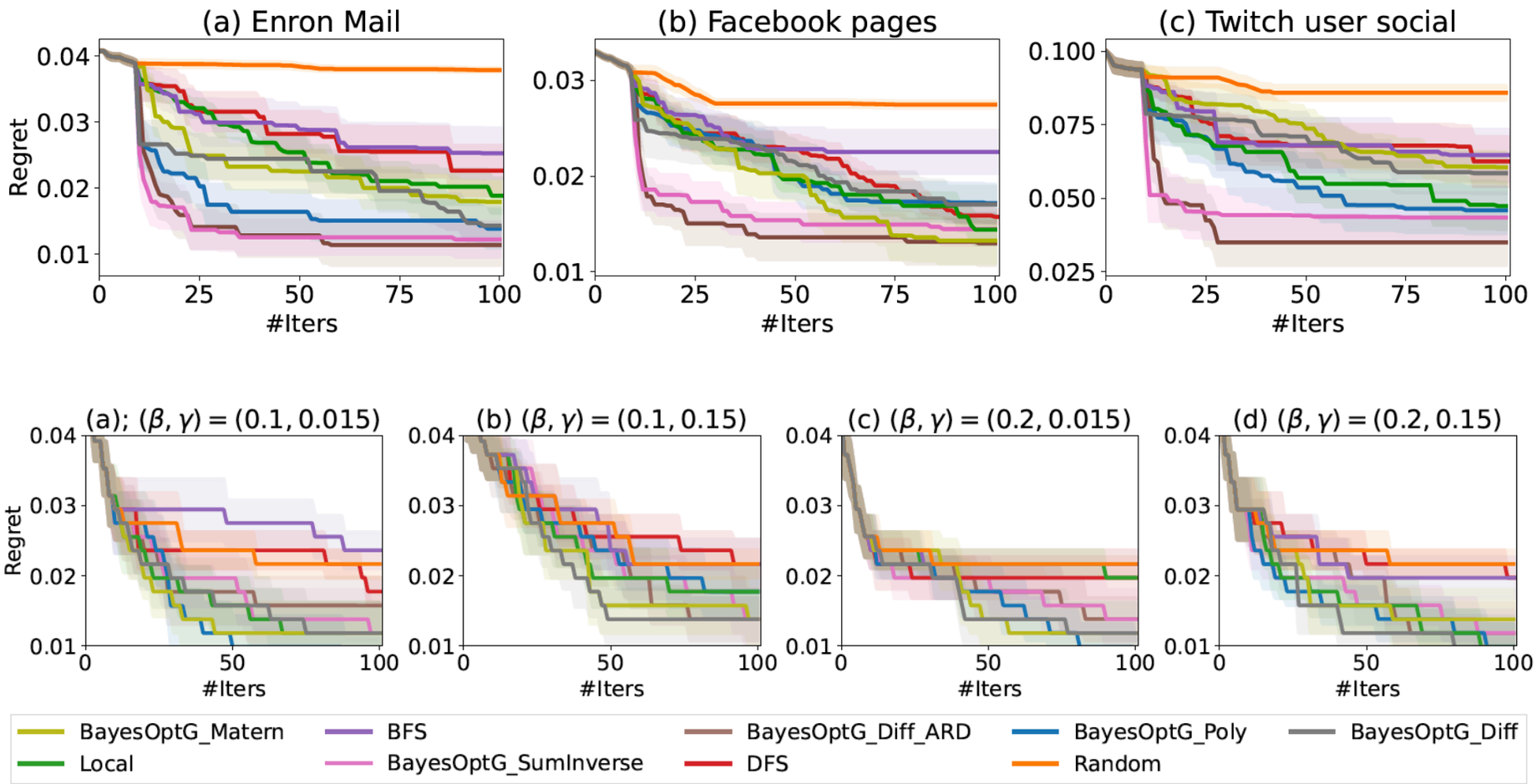
Case 3: Identifying important nodes

- Methodology
  - progressively sample subgraph on-the-fly
  - surrogate model: Gaussian process via kernels measuring **similarity between nodes**
  - acquisition function: expected improvement

Kernel	Regularisation function $r(\lambda_i)$	Kernel function $K(\mathcal{V}, \mathcal{V})$
Diffusion† [37, 27]	$\exp(\beta_i \lambda_i)$	$\sum_{i=1}^{\tilde{n}} \exp(-\beta_i \lambda_i) \mathbf{u}_i \mathbf{u}_i^\top$
Polynomial*	$\sum_{\alpha=0}^{\eta-1} \beta_\alpha \lambda_i^\alpha + \epsilon$	$\sum_{i=1}^{\tilde{n}} \left(\sum_{\alpha=0}^{\eta-1} \beta_\alpha \lambda_i^\alpha + \epsilon \right)^{-1} \mathbf{u}_i \mathbf{u}_i^\top$
Sum-of-inverse polynomials*	$\left(\sum_{\alpha=0}^{\eta-1} \frac{1}{\beta_\alpha \lambda_i^\alpha + \epsilon} \right)^{-1}$	$\sum_{i=1}^{\tilde{n}} \left(\sum_{\alpha=0}^{\eta-1} \frac{1}{\beta_\alpha \lambda_i^\alpha + \epsilon} \right) \mathbf{u}_i \mathbf{u}_i^\top$
Matérn [4]	$\left(\beta\nu + \lambda_i \right)^{\nu}$	$\sum_{i=1}^{\tilde{n}} \left(\beta\nu + \lambda_i \right)^{-\nu} \mathbf{u}_i \mathbf{u}_i^\top$

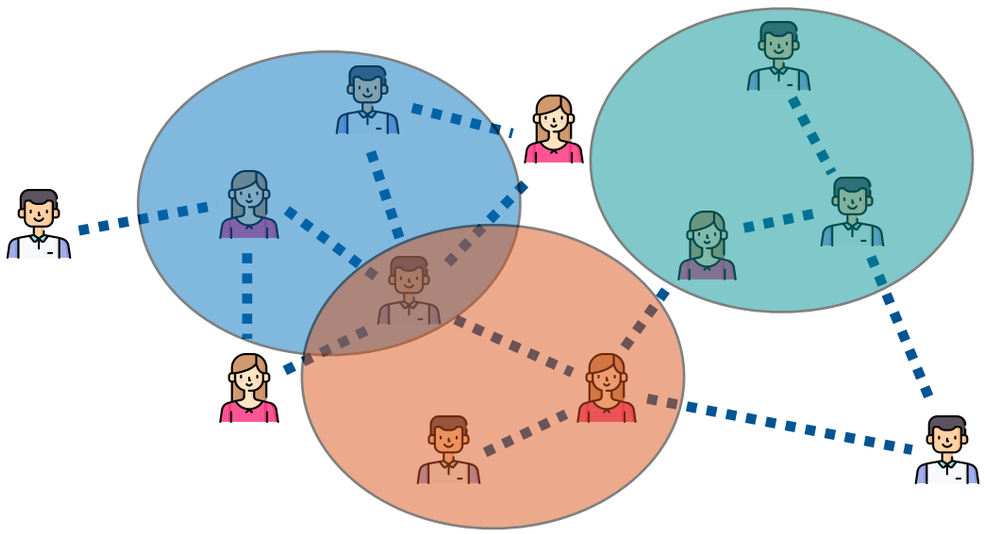
Wan et al., "Bayesian optimisation of functions on graphs," NeurIPS, 2023.

Case 3: Identifying important nodes



Case 4: Identifying important node subsets

- Objective: find node subset with maximum value of function

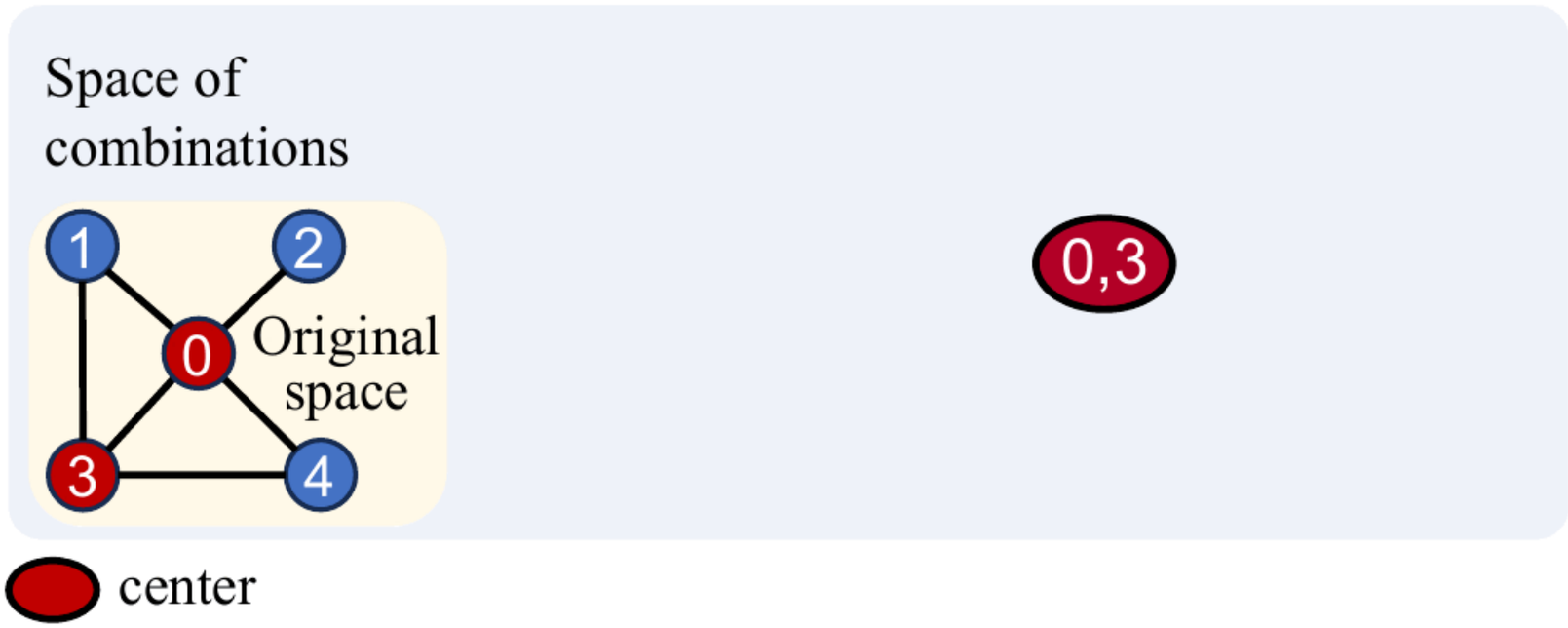


$$\mathcal{S}^* = \arg\max_{\mathcal{S} \in \binom{\mathcal{V}}{k}} f(\mathcal{S})$$

Liang et al., "Bayesian optimization of functions over node subsets in graphs," NeurIPS, 2024.

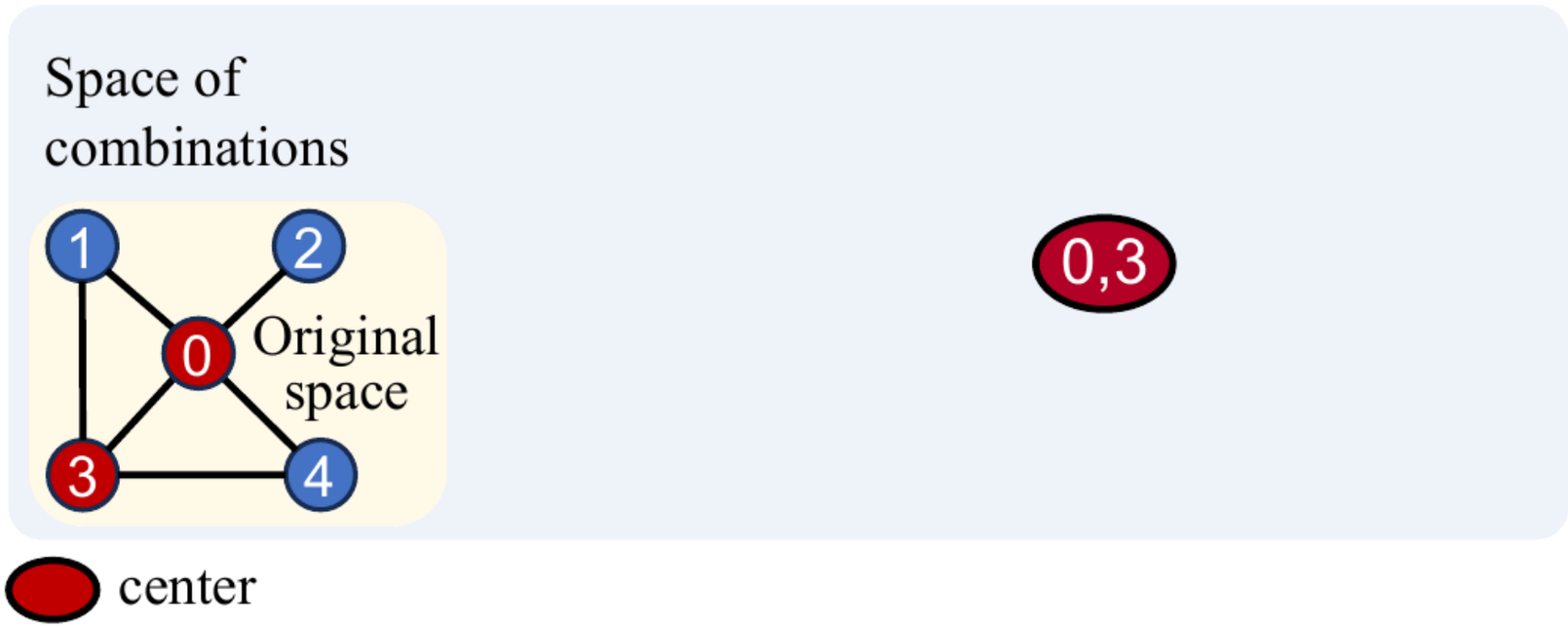
Case 4: Identifying important node subsets

- Methodology
  - convert original graph into combo-graph where each node corresponds to a subset
  - follow methodology in case 3



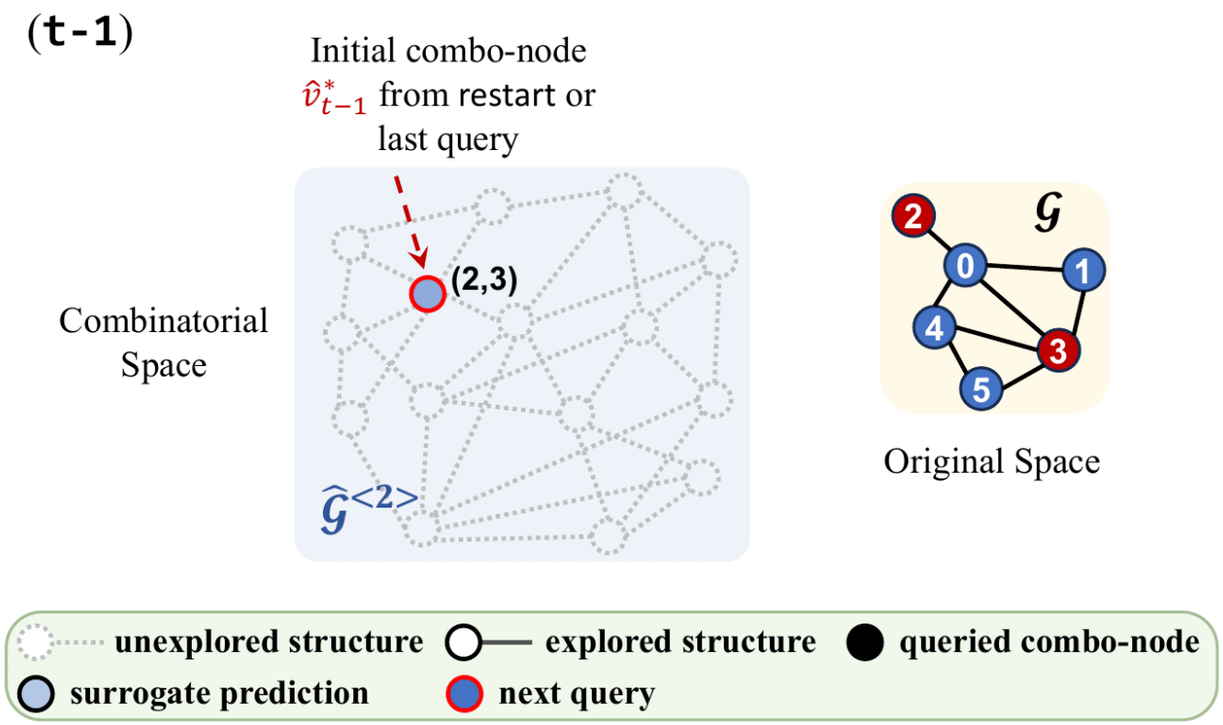
Case 4: Identifying important node subsets

- Methodology
  - convert original graph into combo-graph where each node corresponds to a subset
  - follow methodology in case 3

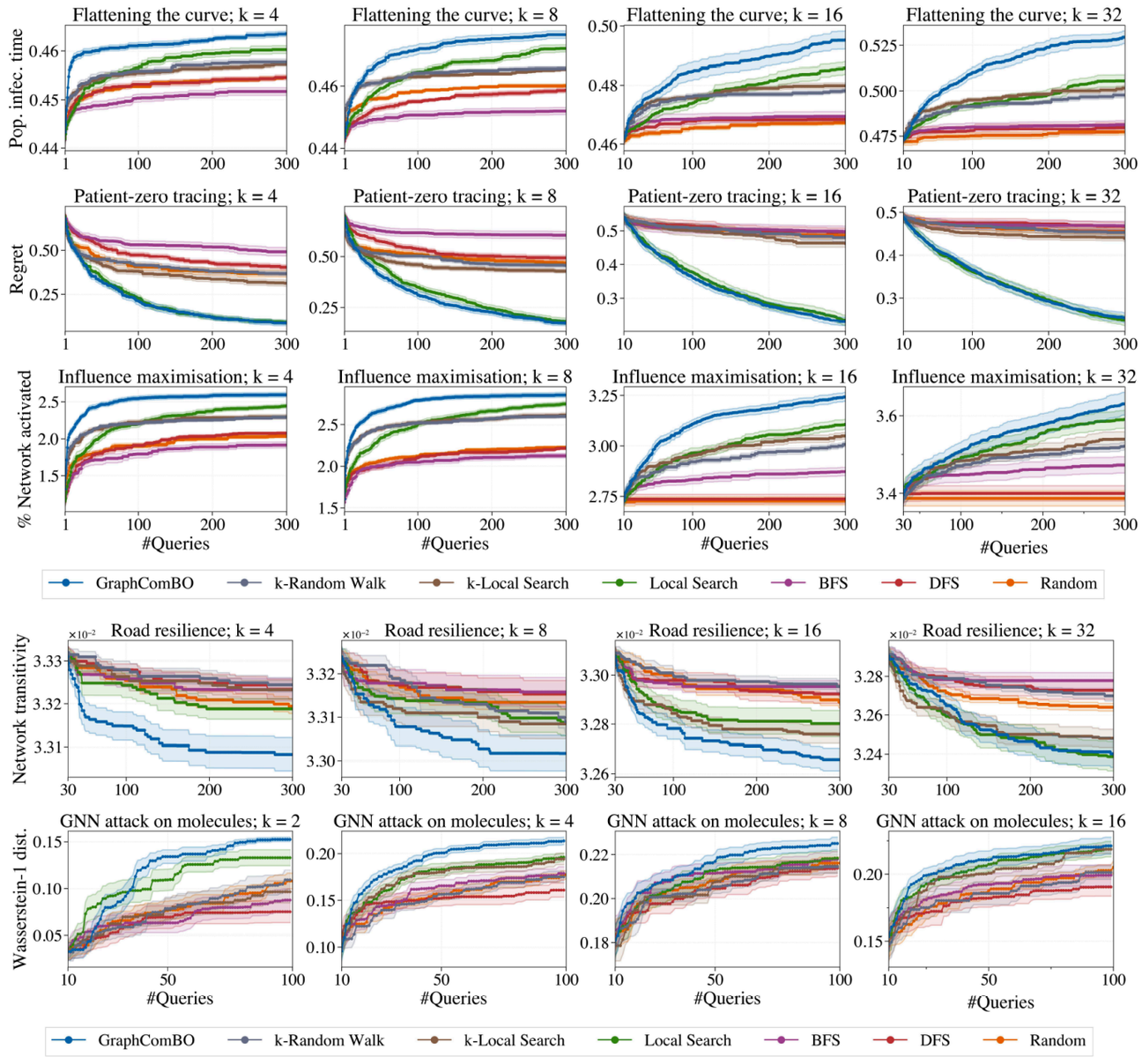


Case 4: Identifying important node subsets

- Methodology
  - convert original graph into combo-graph where each node corresponds to a subset
  - follow methodology in case 3



Case 4: Identifying important node subsets



Lecture 4

- Gaussian processes on graphs
- Bayesian optimisation of graph-based functions
- Discussion

Discussion

- Combination of Bayesian and graph modelling
  - Bayesian modelling provides probabilistic perspective
  - graph modelling provides geometric perspective
- New and interesting problems
  - Gaussian processes/uncertainty estimation on graphs
  - optimisation of graph-based functions
  - optimisation of graph structure for downstream tasks
- Open challenges
  - computational complexity and scalability
  - theoretical analysis

References

- Y.-C. Zhi, Y. C. Ng and X. Dong, "Gaussian processes on graphs via spectral kernel learning," *IEEE TSIPN*, 2023.
- F. L. Opolka, Y.-C. Zhi, P. Liò and X. Dong, "Adaptive Gaussian processes on graphs via spectral graph wavelets," *AISTATS*, 2022.
- B. Ru, X Wan, X. Dong and M. A. Osborne, "Interpretable neural architecture search via Bayesian optimisation with Weisfeiler-Lehman kernels," *ICLR*, 2021.
- X. Wan, H. Kenlay, B. Ru, A. Blaas, M. A. Osborne and X. Dong, "Adversarial attacks on graph classifiers via Bayesian optimisation," *NeurIPS*, 2021.
- X. Wan, P. Osselin, H. Kenlay, B. Ru, M. A. Osborne and X. Dong, "Bayesian optimisation of functions on graphs," *NeurIPS*, 2023.
- H. Liang, X. Wan and X. Dong, "Bayesian optimization of functions over node subsets in graphs," *NeurIPS*, 2024.