Gaussian process regression for high dimensional graph inputs

Raphaël Carpintero Perez
Sébastien Da Veiga
Josselin Garnier
Brian Staber
Introduction

Graph kernels

Sliced Wasserstein Weisfeiler Lehman (SWWL)

Conclusion and future work
Introduction
Objectives
Objectives

Gaussian Process regression

Inputs

Prediction

Uncertainty quantification

Pressure prediction

Standard deviation pressure field

1.74e-02 0.1 0.16 0.222 3.87e+01
Inputs and outputs

- **Graph inputs**
  - Mesh $\rightarrow$ Graph structure
  - 3D coordinates for all nodes

- **Scalar inputs**
  - Pressure
  - Speed of rotation

- **Scalar outputs**
  - Physical quantities
Gaussian process regression

Prior

- random samples
- $\mu$
- $\mu \pm 2\sigma$

Posterior

- random samples
- $\hat{m}$
- $\hat{m} \pm 2\hat{\Sigma}$
- train points
Gaussian process regression

- \( X = (G_1, \cdots, G_N)^T \) with \( G_i \in \Gamma \) (train input graphs)
- \( Y = (y_1, \cdots, y_N)^T \), \( y_i \in \mathbb{R} \) (scalar outputs)

- Observations: \( y_i = f(G_i) + \epsilon_i \) where \( \epsilon_i \sim \mathcal{N}(0, \sigma^2) \)
  \( f: \Gamma \rightarrow \mathbb{R} \)
  \( \bar{f} = (f(G_1), \cdots, f(G_N))^T \)

- Gaussian prior over functions: \( \bar{f} | G_1, \cdots, G_N \sim \mathcal{N}(0, K_{ff}) \)

- \( K_{ff}: N \times N \) covariance matrix where \( K_{ij}^{ff} = k(G_i, G_j) \)
  and \( k: \Gamma \times \Gamma \rightarrow \mathbb{R} \) is a positive definite kernel

- Question: how to choose \( k \)?
What is a graph?
What is a graph?

Case 1: Vertices + Edges

Case 2: Vertices + Edges + Node labels

Case 3: Vertices + Edges + Node attributes

∈ ℝ^s
What is a graph?

Case 1: Vertices + Edges

Case 2: Vertices + Edges + Node labels

Case 3: Vertices + Edges + Node attributes

Case 3A: Fixed structure -> signal

Case 3B: Fixed number of nodes

Case 3C: Varying number of nodes + structure + attributes
What is a graph?

Case 1:
Vertices + Edges

Case 2:
Vertices + Edges + Node labels

Case 3:
Vertices + Edges + Node attributes

Case 3A: Fixed structure -> signal

Case 3B: Fixed number of nodes

Case 3C: Varying number of nodes + structure + attributes + large-scale + sparse
Graph kernels
Invariants / Topological descriptors

- Map the graph to a vectorial representation

- Invariants: do not change under graph isomorphism (diameter, average clustering coefficient, ...)

- Complete invariants require exponential time
Graph edit distance

- $d(G_1, G_2) = \text{minimal number of operations to transform } G_1 \text{ in } G_2$
  (adding/removing an edge/vertex, node relabeling)

- NP-complete

- Not suited for node-attributed graphs...
Taxonomy of graph kernels

Figure from [Nikolentzos et al., 2021]
\( R \)-convolution kernels

- \( R(g_1, \ldots, g_d, G) \): \( R \)-decomposition where \( g_i \) is a ‘part’ of \( G \) (relationship)

- \( R^{-1}(G) = \{ g := (g_1, \ldots, g_d) \mid R(g_1, \ldots, g_d, G) \} \): pre-image of the relation

- Let \( k_i \) a base kernel based on a subset of the parts denoted \( G_i \).

- The \( R \)-convolution kernel between \( G \) and \( G' \) is defined as

\[
k_R(G, G') := \sum_{g \in R^{-1}(G)} \sum_{g' \in R^{-1}(G')} \prod_{i=1}^{d} k_i(g_i, g'_i)
\]
All node-pairs kernel / node histogram kernel

\[ k_N(G, G') := \sum_{v \in V} \sum_{v' \in V'} k_{\text{node}}(v, v') \] where \( k_{\text{node}} \) is a positive definite kernel between node attributes/labels \( \rightarrow \) feature map \( \phi_{\text{node}} \)

- \( k_N(G, G') = \langle \phi_N(G), \phi_N(G') \rangle_{\mathcal{H}} \) where \( \phi_N(G) := \sum_{v \in V} \phi_{\text{node}}(v) \)

- When \( \phi_{\text{node}}(v) = e_{l(v)} \) (\( k_{\text{node}} \) is a Dirac kernel on node labels), \( \phi_N \) is an unnormalized histogram that counts occurrences of node labels
Graphlet kernel

- Set of $k$-graphlets of size $N_k$, $k \geq 3$

- $k$-spectrum of $G$: vector $\phi_{GL}(G)$ of the frequencies of all graphlets in $G$

- $k_{GL}(G, G') := \phi_{GL}(G)\phi_{GL}(G')^T$

- Issue: does not take into account labels or attributes
Graph Hopper

[Feragen et al., 2013]

\[ \mathcal{P}: \text{set of all shortest paths in } G, \quad |\pi|: \text{discrete length of the path } \pi = (\pi_1, \ldots, \pi_{|\pi|}) \]

- Complexity: \( O(n^2(|E| + \log n)) \)

\[
k(G, G') := \sum_{\pi \in \mathcal{P}, \pi' \in \mathcal{P}'} k_p(\pi, \pi') \quad \text{with} \quad k_p(\pi, \pi') := \begin{cases} |\pi| \sum_{j=1}^{RBF(\pi_j, \pi'_j)} & \text{if } |\pi| = |\pi'| \\ 0 & \text{otherwise} \end{cases}
\]
Graph Hopper

[Feragen et al., 2013]

\[ \mathcal{P} \text{: set of all shortest paths in } G, \quad |\pi|: \text{discrete length of the path } \pi = (\pi_1, \ldots, \pi_{|\pi|}) \]

- Complexity: \( O(n^2(|E| + \log n)) \)
Graph Hopper

[Feragen et al., 2013]

\[ k(G, G') := \sum_{\pi \in \mathcal{P}, \pi' \in \mathcal{P}'} k_p(\pi, \pi') \quad \text{with} \quad k_p(\pi, \pi') := \begin{cases} |\pi| & \text{if } |\pi| = |\pi'| \\ \sum_{j=1}^{\min(|\pi|, |\pi'|)} RBF(\pi_j, \pi'_j) & \text{otherwise} \end{cases} \]

- \( \mathcal{P} \): set of all shortest paths in \( G \)
- \(|\pi|\): discrete length of the path \( \pi = (\pi_1, \ldots, \pi_{|\pi|}) \)
- Complexity: \( O(n^2(|E| + \log n)) \)
Sliced Wasserstein Weisfeiler Lehman (SWWL)
Node embeddings + Optimal transport approaches
Wasserstein Weisfeiler-Lehman Graph kernel (step 1)

[Togninalli et al., 2019]
Weisfeiler-Lehman embeddings

Figure From [Kriege et al., 2020]

- WL relabeling (discrete case)

\[ \Sigma = \{A, B\} \]
\[ \Sigma = \{A, B, C, D, E\} \]
\[ \Sigma = \{A, B, C, D, E, F, G, H, I\} \]

\[ l^{(i+1)}(v) = Hash(l^i(v), \{l^i(u), u \in N(v)\}) \]
\[ X_G^{(i)} = [l^{(i)}(v), v \in V_G] \]
\[ X_G = Concatenate(X_G^{(0)}, \ldots, X_G^{(H)}) \]
Continuous Weisfeiler-Lehman embeddings

[Togninalli et al., 2019]

- WL relabeling (continuous case)

\[ a^{(i+1)}(v) = \frac{1}{2} a^{(i)}(v) + \frac{1}{\deg(v)} \sum_{u \in N(v)} w(v, u) a^{(i)}(u) \]

\[ X^{(i)}_G = [a^{(i)}(v), v \in V_G] \quad X_G = \text{Concatenate}(X^{(0)}_G, \ldots, X^{(H)}_G) \]
Wasserstein Weisfeiler-Lehman graph kernel (step 2)
Wasserstein Weisfeiler-Lehman graph kernel (step 2)

\[ \mathcal{E}_G \]

\[ \mathcal{E}_{G'} \]

Wasserstein distance
Wasserstein distance

- \( \forall r \in [1, +\infty), \ P_r(\mathbb{R}^s) \): probability measures on \( \mathbb{R}^s \) with finite moments of order \( r \).

\[
\forall \mu, \nu \in P_r(\mathbb{R}^s), \ W^r_r(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int_{\mathbb{R}^s \times \mathbb{R}^s} ||x - y||^r \, d\pi(x, y)
\]

where:
- \( ||.|| \) denotes the Euclidean norm,
- \( \Pi(\mu, \nu) \) the set of probability measures on \( \mathbb{R}^s \times \mathbb{R}^s \) whose marginals w.r.t. the 1st/2nd variable are resp. \( \mu \) and \( \nu \)

- Discrete case: \( \mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \) \( \nu = \frac{1}{n'} \sum_{i=1}^{n'} \delta_{y_i} \)
Wasserstein distance: issues

- Impossible to build a positive definite kernel (*in dimension ≥ 2 *)  
  
  [Peyré, Cuturi, 2019]

- Computationally expensive : $O(n^3 \log(n))$

- Use case: 1000 graphs with 30 000 vertices
  → **400 days** to build the Gram matrix...
Sliced Wasserstein Weisfeiler Lehman graph kernel

Idea: replace Wasserstein by \textbf{sliced Wasserstein}!

\[ \rightarrow \quad \checkmark \, O(n \log(n)) \quad \text{and} \quad \checkmark \, \text{positive definite} \text{ substitution kernels} \]  

[Meunier et al., 2022]
The **sliced Wasserstein** distance is defined as:

\[
SW^r_d(\mu, \nu) = \int_{S^{d-1}} W^r_d(\theta^*_\#\mu, \theta^*_\#\nu) \, d\sigma(\theta)
\]

where
- \(S^d\) : \(d\)-dimensional unit sphere, \(\sigma\) : uniform distribution on \(S^d\)
- \(\theta^*_\#\mu\) : push-forward measure of \(\mu \in \mathcal{P}_r(\mathbb{R}^s)\) by \(\theta^*_\mathbb{R}^s \rightarrow \mathbb{R}\)

\[
W^r_d(\mu, \nu) = \int_{0}^{1} |F^{-1}(\mu) - F^{-1}(\nu)|^r \, dt
\]

1-d Wasserstein distances between \(\mu\) and \(\nu\).
Sliced Wasserstein distance

- The **sliced Wasserstein** distance is defined as:

\[
SW^r_r(\mu, \nu) = \int_{\mathbb{S}^{d-1}} W_r^r(\theta^*_\#\mu, \theta^*_\#\nu) d\sigma(\theta)
\]

where

- \(\mathbb{S}^d\) : \(d\)-dimensional unit sphere, \(\sigma\) : uniform distribution on \(\mathbb{S}^d\)
- \(\theta^*_\#\mu\) : push-forward measure of \(\mu \in \mathcal{P}_r(\mathbb{R}^s)\) by \(\theta^* \left( \mathbb{R}^s \rightarrow \mathbb{R}, x \mapsto \langle \theta, x \rangle \right)\)

\[
W_r^r(\mu, \nu) = \frac{1}{n} \sum_{i=1}^{n} |x(i) - y(i)|^r
\]

1-d Wasserstein distances between

\[
\mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \quad \text{and} \quad \nu = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i}
\]
Sliced Wasserstein distance

- The **sliced Wasserstein** distance is defined as:

\[ SW^r_r(\mu, \nu) = \int_{\mathbb{S}^{d-1}} W^r_r(\theta_#\mu, \theta_#\nu) d\sigma(\theta) \]

where
- \( \mathbb{S}^d \): \( d \)-dimensional unit sphere, \( \sigma \): uniform distribution on \( \mathbb{S}^d \)
- \( \theta_#\mu \): push-forward measure of \( \mu \in \mathcal{P}_r(\mathbb{R}^s) \) by \( \theta^* \left( \mathbb{R}^s \rightarrow \mathbb{R} \right) \)

\[ W^r_r(\mu, \nu) = \frac{1}{n} \sum_{i=1}^{n} |x(i) - y(i)|^r \]

1-d Wasserstein distances between

\[ \mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \quad \text{and} \quad \nu = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i} \]
Sliced Wasserstein distance

- The sliced Wasserstein distance is defined as:

\[
SW_r^r(\mu, \nu) = \int_{S^{d-1}} W_r^r(\theta_\# \mu, \theta_\# \nu) d\sigma(\theta)
\]

where

- \( S^d \): \( d \)-dimensional unit sphere, \( \sigma \): uniform distribution on \( S^d \)
- \( \theta_\# \mu \): push-forward measure of \( \mu \in \mathcal{P}_r(\mathbb{R}^s) \) by \( \theta^* \left( \mathbb{R}^s \to \mathbb{R} \right) \)

\[
\bar{W}_r^r(\mu, \nu) = \frac{1}{Q} \sum_{q=1}^{Q} |x_{(q)} - y_{(q)}|^r
\]

(Approximation with \( Q \ll \max(n, n') \) quantiles)

1-d Wasserstein distances between

\[
\mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \quad \text{and} \quad \nu = \frac{1}{n'} \sum_{i=1}^{n'} \delta_{y_i}
\]
Sliced Wasserstein distance

- The (estimated) **sliced Wasserstein** distance is defined as:

\[
\widehat{SW}_r^r(\mu, \nu) = \frac{1}{P} \sum_{p=1}^{P} \widehat{W}_r^r(\theta_p^* \# \mu, \theta_p^* \# \nu)
\]

where

- \( S^d \): \( d \)-dimensional unit sphere, \( \sigma \): uniform distribution on \( S^d \)
- \( \theta^*_\# \mu \): push-forward measure of \( \mu \in \mathcal{P}_r(\mathbb{R}^s) \) by \( \theta^* \left( \mathbb{R}^s \rightarrow \mathbb{R} \right) \)

\[
\widehat{W}_r^r(\mu, \nu) = \frac{1}{Q} \sum_{q=1}^{Q} |x_{(q)} - y_{(q)}|^r
\]

(Approximation with \( Q \ll \max(n, n') \) quantiles)

1-d Wasserstein distances between

\[
\mu = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \quad \text{and} \quad \nu = \frac{1}{n'} \sum_{i=1}^{n'} \delta_{y_i}
\]
Sliced Wasserstein Weisfeiler Lehman (SWWL)
Sliced Wasserstein Weisfeiler Lehman (SWWL)

\[ \phi: G \mapsto X_G \in \mathbb{R}^{|V_G| \times d(H+1)} \text{ : WL embeddings after H iterations} \]

\[ k_{SWWL}(G, G') = e^{-\lambda \mathcal{S}\mathcal{W}_2^2(\phi(G), \phi(G'))} \quad (* \text{considering by abuse \( \phi(G), \phi(G') \) as empirical measures \*) \]

with

\[ \mathcal{S}\mathcal{W}_2^2(\mu, \nu) = \frac{1}{PQ} \sum_{p=1}^{P} \sum_{q=1}^{Q} \left| u_{\theta_p}^q - u'_{\theta_q}^q \right|^2 = \left\| E_{\phi(G)} - E_{\phi(G')} \right\|_2^2 \]

\[ \rightarrow \text{Precomputed embeddings } E_{\phi(G)}, E_{\phi(G')} \in \mathbb{R}^{PQ} \text{ where } u_{\theta_p}^q = \langle \theta_p, \phi(G) \rangle_{(q)} \]

\[ E_{\phi(G)} = [u_1^{\theta_1}, \ldots, u_Q^{\theta_1}, \ldots, u_1^{\theta_P}, \ldots, u_Q^{\theta_P}] \]

\[ \text{Complexity for the Gram matrix (sparse graphs):} \]

\[ O(NHn) + NP n \log n + N^2 PQ \]

WL iterations Quantiles Usual RBF kernel
SWWL: experiments on meshes

RMSE (5 exp)

Time to build the Gram matrix

(*) in parallel, using 100 jobs

<table>
<thead>
<tr>
<th>Kernel/Dataset</th>
<th>Rotor37</th>
<th>Rotor37-CM</th>
<th>Tensile2d</th>
<th>Tensile2d-CM</th>
<th>AirfRANS</th>
<th>AirfRANS-CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWWL</td>
<td>1.44 ± 0.07</td>
<td>3.49 ± 0.15</td>
<td>0.89 ± 0.01</td>
<td>1.51 ± 0.01</td>
<td>7.56 ± 0.36</td>
<td>9.63 ± 0.54</td>
</tr>
<tr>
<td>WWL</td>
<td>-</td>
<td>3.51 ± 0.00</td>
<td>-</td>
<td>6.46 ± 0.00</td>
<td>-</td>
<td>14.4 ± 0.80</td>
</tr>
<tr>
<td>PK</td>
<td>-</td>
<td>4.18 ± 0.39</td>
<td>-</td>
<td>6.03 ± 4.58</td>
<td>-</td>
<td>8.94 ± 2.31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel/Dataset</th>
<th>Rotor37</th>
<th>Rotor37-CM</th>
<th>Tensile2d</th>
<th>Tensile2d-CM</th>
<th>AirfRANS</th>
<th>AirfRANS-CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWWL</td>
<td>1min + 11s</td>
<td>4s + 11s</td>
<td>11s + 4s</td>
<td>2s + 4s</td>
<td>5min + 7s</td>
<td>15s + 7s</td>
</tr>
<tr>
<td>WWL</td>
<td>-</td>
<td>13min (*)</td>
<td>-</td>
<td>6min (*)</td>
<td>-</td>
<td>8h (*)</td>
</tr>
<tr>
<td>PK</td>
<td>-</td>
<td>1min</td>
<td>-</td>
<td>2min</td>
<td>-</td>
<td>15min</td>
</tr>
</tbody>
</table>
Conclusion and future work
Conclusion

- Limits of existing graph kernels
  - Many do not handle continuous attributes
  - Many do not scale well to large graphs
  - Many do not guarantee positive definiteness
  - Many are too dependent on the graph structure

- We propose the Sliced Wasserstein Weisfeiler Lehman (SWWL) kernel
  - Positive definite
  - Tractable for large graphs
  - Competitive results for mesh-based Gaussian process regression

- Future work
  - Extension to multiple outputs (e.g. vector fields)
References

- **Graph kernels, Gaussian processes**

- **Optimal transport**
Acknowledgments

- This work was supported by the French National Research Agency (ANR) through the SAMOURAI project under grant ANR20-CE46-0013.

ANR © SAMOURAI
Other approaches using Optimal Transport
Other approaches

- Many approaches with GCNNs and message passing layers
  → Continuous WL of torch_geometric

- Other node embedding: $a^{(i+1)}(v) = \sum_{u \in N(v) \cup \{v\}} \frac{w(v, u)}{\sqrt{\deg(u) \deg(v)}} a^{(i)}(u)$

- Wasserstein embeddings with Linear Optimal transport  [Kolouri et al., 2020]
- Pooling by Sliced-Wasserstein (PSWE)                          [Naderializadeh., 2021]

- Template-based GNN with OT                                   [Vincent-Cuaz et al., 2022]
Wasserstein embeddings

[Kolouri et al., 2020]

- Linear Wasserstein embedding (Linear Optimal transport LOT Framework)
- Transport displacements from a reference distribution to node embeddings
**Wasserstein embeddings**

[Kolouri et al., 2020]

- Given a first node embedding $\phi: G \mapsto X_G \in \mathbb{R}^{|V_G| \times s}$

- $X_0 \in \mathbb{R}^{n_0 \times s}$ reference node embedding

- Linear Wasserstein embedding:
  $\psi_0(X_G) := (u_{G,0} - Id)\sqrt{n_0}$
  where $u_{G,0}$ is the Monge map that pushes $X_0$ to $X_G$

- New graph embedding: $\psi(G) := \psi_0(\phi(G)) \in \mathbb{R}^{n_0 \times s}$ of fixed size

- **Only $N$ Monge map calculations needed**

- Choice of the reference embedding? (Not clear)
Fused Gromov-Wasserstein distance

[Vayer et al., 2019]

- $G = (V_G, E_G, l_a, l_s)$ with $l_a : V_G \rightarrow \mathbb{R}^3$ the coordinate function
- $l_s : V_G \rightarrow \Omega_G$ with $(\Omega_G, c_G)$ a metric space dependant of $G$
- $c_G : \Omega_G \times \Omega_G \rightarrow \mathbb{R}_+ \text{ ‘similarity’ of points in } G$
  (structure-dependent)
  e.g.: $c_G(l_s(v_1), l_s(v_2)) = d_{PCC}(v_1, v_2|G)$
- $a_i = l_a(v_i), s_i = l_s(v_i)$: attributes/structure of point $i$
- $\mu_G = \sum_{i=1}^{n_G} \frac{1}{n_G} \delta(a_i,s_i)$: measure of $G$
- $C_G = [c_G(s_i, s_j)]_{1 \leq i, j \leq n_G}, C_G' = [c_G'(s'_i, s'_j)]_{1 \leq i, j \leq n_G'}$
Fused Gromov-Wasserstein distance

[Vayer et al., 2019]

- $L_{G,G'} = |C_G[i,k] - C_{G'}[j,l]|_{i,j,k,l} \in \mathbb{R}^{n_G \times n_{G'} \times n_G \times n_{G'}}$
- $M_{G,G'} = \left[ ||a_i - a'_j||_2 \right]_{1 \leq i \leq n_G} \in \mathbb{R}^{n_G \times n_{G'}}$
- $FGW_{q,\alpha}(\mu_G,\mu_{G'}) = \min_{\pi \in \Pi} \left( aM_{G,G'}^q + (1 - \alpha)L_{G,G'}^q \otimes \pi, \pi \right)$

Wasserstein Gromov-Wasserstein

- **Issue:** $k(G, G') = e^{-\gamma FGW_{q,\alpha}(\mu_G,\mu_{G'})}$ is not positive definite
Template based GNN with OT

[Vincent-Cuaz et al., 2022]
Graph Convolutional Gaussian Processes

[Walker et al., 2019]

- Graph Convolutional Gaussian Processes
- Local patches around vertices are defined using Spatial-domain charting

- \( J \): number of bins
- Convolution operator on the graph signal \( \psi: V \rightarrow \mathbb{R}^3 \):
  \[
  D_j(v) \psi = \sum_{u \in V} \psi(u) u_j(u, v) \quad \forall j \in \{1, \ldots, J\}
  \]
- \( u_j \): geodesic polar weighting function e.g.
Future work: Anisotropic SWWL?
Future work: Anisotropic SWWL?
Future work: Anisotropic SWWL?

Anisotropic SWWL:

\[
\phi^{(i)} : G \mapsto X_G^{(i)} \in \mathbb{R}^{|V_G| \times d} \quad (i\text{-th iteration of WL})
\]

\[
k_{ASWWL}(G, G') = e^{-\sum_{i=0}^{H} \lambda_i \text{SW}_2^2(\phi^{(i)}(G), \phi^{(i)}(G'))}
\]