Learning node representations using stationary flow prediction on large payment and cash transaction networks

Ciwan Ceylan[1,2] Salla Franzén[2] Florian T. Pokorny[1]

[1]RPL, EECS, KTH Royal Institute of Technology [2]SEB Group

ICML 2021

Transaction data

- Vast number of payments and cash transactions executed daily
- This data is useful for business intelligence and financial crime prevention

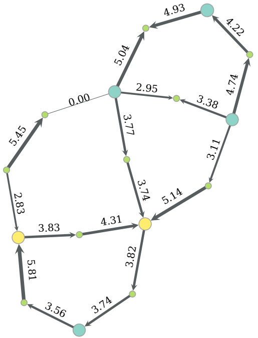
Row	block_timestamp	from_address	to_address	amount
1	2018-06-01 07:06:17 UTC	0x8e5d083f	0x30727e88	150000.0
2	2018-06-01 07:12:35 UTC	0x30727e88	0x876eabf4	149999.0
3	2018-06-01 07:16:15 UTC	0x876eabf4	0x742d35cc	135130.0
4	2018-06-06 06:53:46 UTC	0x8e5d083f	0x052af704	259457.0
5	2018-06-08 06:33:24 UTC	0x53a46c10	0x6f0881f9	188087.0
6	2018-06-12 16:42:34 UTC	0x052af704	0x36abd00f	150000.0

Transaction data

- Vast number of payments and cash transactions executed daily
- This data is useful for business intelligence and financial crime prevention
- Time-aggregated transactions result in a graph with a flow

Row	block_timestamp	from_address	to_address	amount
1	2018-06-01 07:06:17 UTC	0x8e5d083f	0x30727e88	150000.0
2	2018-06-01 07:12:35 UTC	0x30727e88	0x876eabf4	149999.0
3	2018-06-01 07:16:15 UTC	0x876eabf4	0x742d35cc	135130.0
4	2018-06-06 06:53:46 UTC	0x8e5d083f	0x052af704	259457.0
5	2018-06-08 06:33:24 UTC	0x53a46c10	0x6f0881f9	188087.0
6	2018-06-12 16:42:34 UTC	0x052af704	0x36abd00f	150000.0

Aggregate → over time



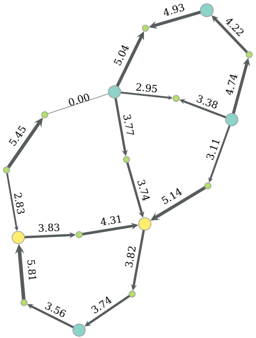
Transaction data

- Vast number of payments and cash transactions executed daily
- This data is useful for business intelligence and financial crime prevention
- Time-aggregated transactions result in a graph with a flow
- How can node representation be learned from the flow?

Row	block_timestamp	from_address	to_address	amount
1	2018-06-01 07:06:17 UTC	0x8e5d083f	0x30727e88	150000.0
2	2018-06-01 07:12:35 UTC	0x30727e88	0x876eabf4	149999.0
3	2018-06-01 07:16:15 UTC	0x876eabf4	0x742d35cc	135130.0
4	2018-06-06 06:53:46 UTC	0x8e5d083f	0x052af704	259457.0
5	2018-06-08 06:33:24 UTC	0x53a46c10	0x6f0881f9	188087.0
6	2018-06-12 16:42:34 UTC	0x052af704	0x36abd00f	150000.0

Aggregate

over time

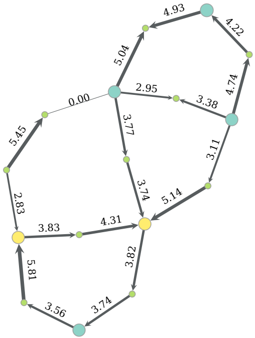


Transaction data

- Vast number of payments and cash transactions executed daily
- This data is useful for business intelligence and financial crime prevention
- Time-aggregated transactions result in a graph with a flow
- How can node representation be learned from the flow?
- Dataset used: 1.5 years of publicly available ethereum transactions

Row	block_timestamp	from_address	to_address	amount
1	2018-06-01 07:06:17 UTC	0x8e5d083f	0x30727e88	150000.0
2	2018-06-01 07:12:35 UTC	0x30727e88	0x876eabf4	149999.0
3	2018-06-01 07:16:15 UTC	0x876eabf4	0x742d35cc	135130.0
4	2018-06-06 06:53:46 UTC	0x8e5d083f	0x052af704	259457.0
5	2018-06-08 06:33:24 UTC	0x53a46c10	0x6f0881f9	188087.0
6	2018-06-12 16:42:34 UTC	0x052af704	0x36abd00f	150000.0

Aggregate over time →



Gradient Flow Model[1]

- Hodge Decomposition of flow $\mathbf{y} \in \mathbb{R}^{|E|}$ on a graph $G = (V, E)$
  - $\mathbf{y} = \mathbf{y}_{\text{grad}} + \mathbf{y}_{\text{div}}$ s.t. $\mathbf{y}_{\text{grad}} \perp \mathbf{y}_{\text{div}}$
  - $y^{(ij)}_{\text{grad}} = z^{(j)} - z^{(i)}$ with $z^{(i)} \in \mathbb{R}$ and $(i, j) \in E$

[1] Lek-Heng Lim. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.

Gradient Flow Model[1]

- Hodge Decomposition of flow $\mathbf{y} \in \mathbb{R}^{|E|}$ on a graph $G = (V, E)$
  - $\mathbf{y} = \mathbf{y}_{\text{grad}} + \mathbf{y}_{\text{div}}$ s.t. $\mathbf{y}_{\text{grad}} \perp \mathbf{y}_{\text{div}}$
  - $y^{(ij)}_{\text{grad}} = z^{(j)} - z^{(i)}$ with $z^{(i)} \in \mathbb{R}$ and $(i, j) \in E$
- Minimize over z: $\sum_{ij \in E} \left(y^{(ij)} - (z^{(j)} - z^{(i)})\right)^2$

[1] Lek-Heng Lim. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.

Gradient Flow Model[1]

- Hodge Decomposition of flow $\mathbf{y} \in \mathbb{R}^{|E|}$ on a graph $G = (V, E)$
  - $\mathbf{y} = \mathbf{y}_{\text{grad}} + \mathbf{y}_{\text{div}}$ s.t. $\mathbf{y}_{\text{grad}} \perp \mathbf{y}_{\text{div}}$
  - $y^{(ij)}_{\text{grad}} = z^{(j)} - z^{(i)}$ with $z^{(i)} \in \mathbb{R}$ and $(i, j) \in E$
- Minimize over z: $\sum_{ij \in E} \left(y^{(ij)} - (z^{(j)} - z^{(i)})\right)^2$
- Gradient flow model: $f^{(ij)}_{\text{grad}} = z^{(j)} - z^{(i)}$

[1] Lek-Heng Lim. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.

Gradient Flow Model[1]

- Hodge Decomposition of flow $\mathbf{y} \in \mathbb{R}^{|E|}$ on a graph $G = (V, E)$
  - $\mathbf{y} = \mathbf{y}_{\text{grad}} + \mathbf{y}_{\text{div}}$ s.t. $\mathbf{y}_{\text{grad}} \perp \mathbf{y}_{\text{div}}$
  - $y_{\text{grad}}^{(ij)} = z^{(j)} - z^{(i)}$ with $z^{(i)} \in \mathbb{R}$ and $(i, j) \in E$
- Minimize over z: $\sum_{ij \in E} \left(y^{(ij)} - (z^{(j)} - z^{(i)})\right)^2$
- Gradient flow model: $f_{\text{grad}}^{(ij)} = z^{(j)} - z^{(i)}$

Two issues:

1. Gradient model cannot learn $\mathbf{y}_{\text{div}}$.
   For ethereum dataset:
   $\| \mathbf{y}_{\text{div}} \|_2 = 42\text{M}$, $\| \mathbf{y}_{\text{grad}} \|_2 = 6\text{M}$

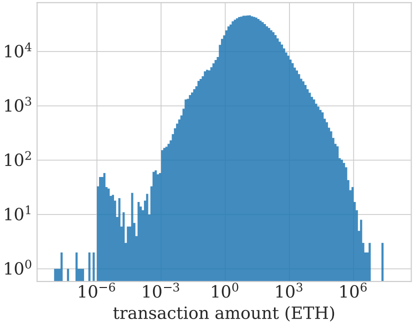
[1] Lek-Heng Lim. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.

Gradient Flow Model[1]

- Hodge Decomposition of flow $\mathbf{y} \in \mathbb{R}^{|E|}$ on a graph $G = (V, E)$
  - $\mathbf{y} = \mathbf{y}_{\text{grad}} + \mathbf{y}_{\text{div}}$ s.t. $\mathbf{y}_{\text{grad}} \perp \mathbf{y}_{\text{div}}$
  - $y_{\text{grad}}^{(ij)} = z^{(j)} - z^{(i)}$ with $z^{(i)} \in \mathbb{R}$ and $(i, j) \in E$
- Minimize over z: $\sum_{ij \in E} \left(y^{(ij)} - (z^{(j)} - z^{(i)})\right)^2$
- Gradient flow model: $f_{\text{grad}}^{(ij)} = z^{(j)} - z^{(i)}$

Two issues:

1. Gradient model cannot learn $\mathbf{y}_{\text{div}}$.
   For ethereum dataset:
   $\| \mathbf{y}_{\text{div}} \|_2 = 42\text{M}$, $\| \mathbf{y}_{\text{grad}} \|_2 = 6\text{M}$
2. Squared error unsuitable for multi-scale and heavy-tail transactions



[1] Lek-Heng Lim. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.

Addressing expressivity

- Gradient model is limited by scalar potentials
  - Use multiple potentials for each node, $\mathbf{z}^{(i)} \in \mathbb{R}^K$
  - Use a gate function $\bar{\sigma} : \mathbb{R}^{K \times K} \mapsto [0, 1]^K$
- Gated gradient flow model: $f^{(ij)} = \bar{\sigma}(\mathbf{u}^{(i)}, \mathbf{u}^{(j)})^T \left(\mathbf{z}^{(j)} - \mathbf{z}^{(i)}\right)$

Addressing expressivity

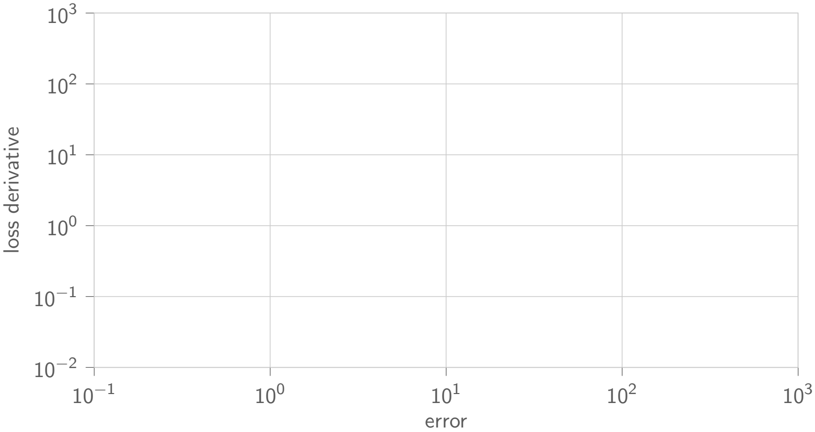
- Gradient model is limited by scalar potentials
  - Use multiple potentials for each node, $\mathbf{z}^{(i)} \in \mathbb{R}^K$
  - Use a gate function $\bar{\sigma} : \mathbb{R}^{K \times K} \mapsto [0,1]^K$
- Gated gradient flow model: $f^{(ij)} = \bar{\sigma}(\mathbf{u}^{(i)}, \mathbf{u}^{(j)})^T \left(\mathbf{z}^{(j)} - \mathbf{z}^{(i)}\right)$
- Theorem: This model can express any flow on a graph, given sufficiently expressive $\bar{\sigma}(\cdot,\cdot)$ and $K = 2\Delta(G)$

Addressing expressivity

- Gradient model is limited by scalar potentials
  - Use multiple potentials for each node, $\mathbf{z}^{(i)} \in \mathbb{R}^K$
  - Use a gate function $\bar{\sigma} : \mathbb{R}^{K \times K} \mapsto [0,1]^K$
- Gated gradient flow model: $f^{(ij)} = \bar{\sigma}(\mathbf{u}^{(i)}, \mathbf{u}^{(j)})^T \left(\mathbf{z}^{(j)} - \mathbf{z}^{(i)}\right)$
- Theorem: This model can express any flow on a graph, given sufficiently expressive $\bar{\sigma}(\cdot,\cdot)$ and $K = 2\Delta(G)$
- We use a simple form for $\bar{\sigma}$ with learnable $\mathbf{u}^{(i)}$:

$$[\bar{\sigma}(\mathbf{u}^{(i)}, \mathbf{u}^{(j)})]_k = \frac{1}{1 + e^{-(u_k^{(i)} + u_k^{(j)})}}$$

Addressing training loss function



Stationary flow prediction

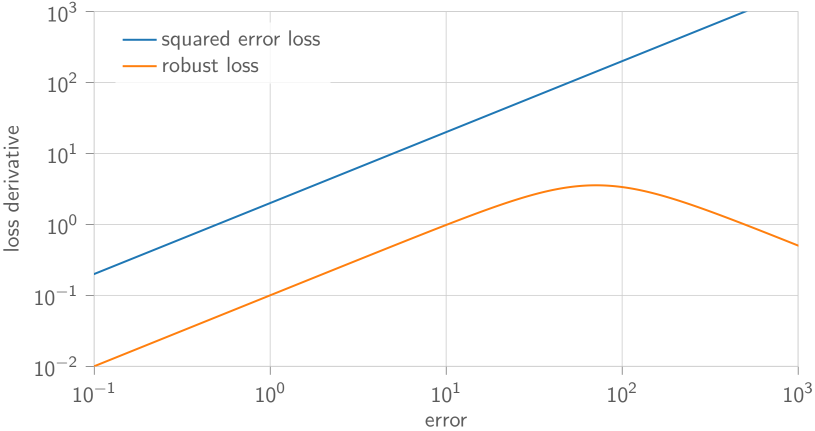
minimize $\sum_{ij \in E} \ell(y^{(ij)}, f^{(ij)})$

Addressing training loss function

Stationary flow prediction

minimize $\sum_{ij \in E} \ell(y^{(ij)}, f^{(ij)})$

Addressing training loss function



Stationary flow prediction

minimize $\sum_{ij \in E} \ell(y^{(ij)}, f^{(ij)})$

Robust loss

$\ell(y^{(ij)}, f^{(ij)}) = \log\left(1 + \frac{(y^{(ij)} - f^{(ij)})^2}{\nu |y^{(ij)}|_\tau}\right)$

Baseline models

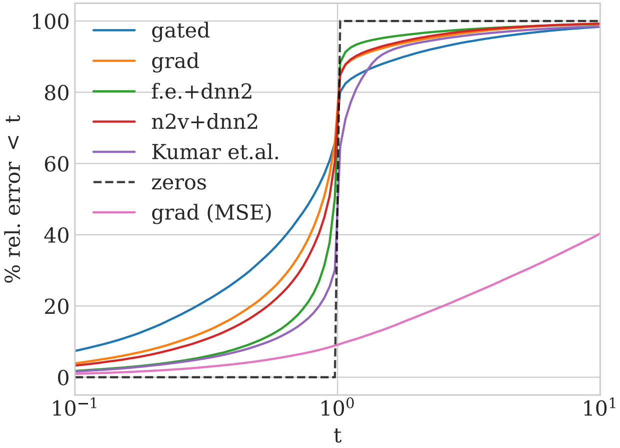
- Gradient model (Robust loss)
- Gradient model, MSE (Squared loss)
- 6 engineered node features + 2 layer MLP (Robust loss)
- 128 node2vec[2] node representations + 2 layer MLP (Robust loss)
- Weight prediction model by Kumar et. al.[3]

[2] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[3] S. Kumar et al. "Edge Weight Prediction in Weighted Signed Networks". In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 2016, pp. 221–230.

Quantitative results

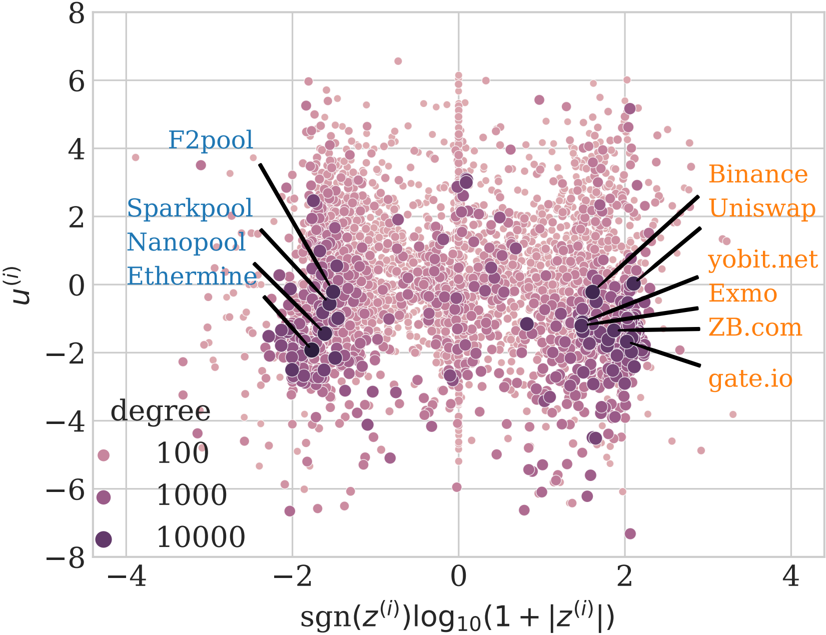
- Rel. error: $|y^{(ij)} - f^{(ij)}|/|y^{(ij)}|$
- Trivial baseline: $f^{(ij)} = 0$



Median abs. error

GATED	GRAD	F.E.+DNN2	N2V+DNN2	KUMAR ET.AL.	ZEROS	GRAD (MSE)
9.34	10.39	13.10	11.25	14.19	14.61	396.63

Qualitative results



Learning node representations using stationary flow prediction on large payment and cash transaction networks

Ciwan Ceylan[1,2] Salla Franzén[2] Florian T. Pokorny[1]

[1]RPL, EECS, KTH Royal Institute of Technology [2]SEB Group

ICML 2021

Thank you for your attention!

References

Grover, Aditya and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

Kumar, S. et al. "Edge Weight Prediction in Weighted Signed Networks". In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 2016, pp. 221–230.

Lim, Lek-Heng. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.