Supplementary Materials for "A simple way to learn metrics between attributed graphs"

Yacouba Kaloga, Pierre Borgnat, Amaury Habrard

A Appendix

A.1 Motivation of $\mathcal{RPW}_2$

We give here, additional information to justify the form that $\mathcal{RPW}_2$ (Eq. (8)) takes. In particular, the choice of using canonical vector basis $\{u_i\}_{i=1}^p$ of $\mathbb{R}^p$ to project distributions. We will show that this choice ensures that $\mathcal{RPW}_2$ is a metric on discrete distribution space of $\mathbb{R}^p$, verifying symmetry, identity of indiscernibles and triangular inequality properties just like $\mathcal{PW}_2$ from which it is derived. Then we will show that for any choice of vector family which does not span $\mathbb{R}^p$, ensuring the identity of indiscernibles may require the same amount of computation as to calculate $\mathcal{W}_2$. Note that here we will not write the proof that $\mathcal{PW}_2$ is a metric since it can be derived straightforwardly from the proof that $\mathcal{RPW}_2$ is a metric. It is also easy to derive from this proof that $\mathcal{RPW}_s$ (and $\mathcal{PW}_s$) for $s \in \mathbb{N}^*$ is also a metric on discrete distribution space.

In order to facilitate the reading we recall below the definition of $\mathcal{RPW}_2$:

$$\mathcal{RPW}_2(\mu,\nu)^2 = \frac{1}{p}\sum_{k=1}^{p}\sum_{i,j=1}^{n,n'} \pi_{i,j}^{\boldsymbol{u}_k,*}\|\boldsymbol{x}_i - \boldsymbol{x}'_j\|_2^2 \tag{12}$$

Where $\mu = \sum_{i=1}^n a_i\delta_{\boldsymbol{x}_i}$ and $\nu = \sum_{i=1}^{n'} b_i\delta_{\boldsymbol{x}'_i}$ are distributions of $\mathbb{R}^p$ with strict positive weights $(a_i)_{i=1}^n$ and $(b_i)_{i=1}^{n'}$ which add up to 1; $\pi^{\boldsymbol{u}_k,*}$ are 1-d optimal transport plans of projected distributions $\mu_{\boldsymbol{u}_k} = \sum_{i=1}^n a_i\delta_{\boldsymbol{x}_i(k)}$ and $\nu_{\boldsymbol{u}_k} = \sum_{i=1}^{n'} b_i\delta_{\boldsymbol{x}'_i(k)}$. Since $\mu_{\boldsymbol{u}_k}$ and $\nu_{\boldsymbol{u}_k}$ are projected distribution they may have non unique bins which lead to potentially several optimal transport plans, in such cases the chosen transport $\pi_{i,j}^{\boldsymbol{u}_k,*}$ is one of those which minimize Eq. (12).

A.1.1 $\mathcal{RPW}_2$ is a metric on discrete distribution space

Let's show that $\mathcal{RPW}_2$ verifies the three canonical properties of metrics.

Symmetry. The symmetry is straightforward to derive. Since the cost $\|\boldsymbol{x}_i - \boldsymbol{x}'_j\|_2^2$ and the transport plans $\pi^{\boldsymbol{u}_k,*}$ (which depend exclusively on the sorting of $\big(x_i(k)\big)_{i\in\{1,\dots,n\}}$ and $\big(x'_j(k)\big)_{j\in\{1,\dots,n'\}}$) are symmetric, $\mathcal{RPW}_2$ is thus symmetric.

Identity of indiscernibles. We have to show that $\mathcal{RPW}_2(\mu,\nu) = 0$ if and only if $\mu = \nu$.

First, let's assume that $\mu \neq \nu$, then:

$$\mathcal{RPW}_2(\mu,\nu)^2 = \frac{1}{p}\sum_{k=1}^{p}\sum_{i,j=1}^{n,n'} \pi_{i,j}^{\boldsymbol{u}_k,*}\|\boldsymbol{x}_i - \boldsymbol{x}'_j\|_2^2 \tag{13}$$

$$\text{By definition of } \mathcal{W}_2 \text{ (Eq. (3))} \qquad \geq \frac{1}{p}\sum_{k=1}^{p}\mathcal{W}_2(\mu,\nu)^2 \tag{14}$$

$$\text{Because } \mathcal{W}_2 \text{ is a metric and } \mu \neq \nu \qquad > 0 \tag{15}$$

Hence $\mathcal{RPW}_2(\mu,\nu) = 0 \implies \mu = \nu$.

Secondly, let's assume that $\mu = \nu$. This implies that $n = n'$, $a_i = b_i$ and $\mu_{\boldsymbol{u}_k} = \nu_{\boldsymbol{u}_k}$ for $i \in \{1,\dots,n\}$ and $k \in \{1,\dots,p\}$. Because of these equalities, each optimal transport $\pi^{\boldsymbol{u}_k,*}$ can be associated to a unique permutation. By introducing the notation σ_k such that $\forall i,j \in \{1,\dots,n\}, \sigma_k(j) = i$ iff $\pi_{i,j}^{\boldsymbol{u}_k,*} = a_j$, we can write:

$$\mathcal{RPW}_2(\mu,\nu)^2 = \frac{1}{p}\sum_{k=1}^{p}\sum_{j=1}^{n} a_j \|\boldsymbol{x}_{\sigma_k(j)} - \boldsymbol{x}'_j\|_2^2 \tag{16}$$

In addition, since $\mu = \nu$, they have the same bins, therefore there is a permutation τ such that:

$$\boldsymbol{x}_{\tau(j)} = \boldsymbol{x}'_j \tag{17}$$

Thus:

$$\forall k \in \{1,\dots,p\},\ \sum_{j=1}^{n} a_j \|\boldsymbol{x}_{\tau(j)}(k) - \boldsymbol{x}'_j(k)\|_2^2 = 0 \tag{18}$$

Since, $\forall k \in \{1,\dots,p\}$:

$$0 = \mathcal{W}_2(\mu_{\boldsymbol{u}_k}, \nu_{\boldsymbol{u}_k}) = \sum_{j=1}^{n} a_i \|\boldsymbol{x}_{\tau(j)}(k) - \boldsymbol{x}'_j(k)\|_2^2 \tag{19}$$

It is clear that τ is a permutation associated to an optimal transport between $\nu_{\boldsymbol{u}_k}$ and $\mu_{\boldsymbol{u}_k}$. Therefore by definition of σ_k (i.e. a permutation associated with the optimal transport between $\nu_{\boldsymbol{u}_k}$ and $\mu_{\boldsymbol{u}_k}$ which minimizes Eq. (12)):

$$\mathcal{RPW}_2(\mu,\nu)^2 = \frac{1}{p}\sum_{k=1}^{p}\sum_{j=1}^{n} a_j \|\boldsymbol{x}_{\sigma_k(j)} - \boldsymbol{x}'_j\|_2^2 \leq \frac{1}{p}\sum_{k=1}^{p}\sum_{j=1}^{n} a_j \|\boldsymbol{x}_{\tau(j)} - \boldsymbol{x}'_j\|_2^2 = 0 \tag{20}$$

Hence $\mu = \nu \implies \mathcal{RPW}_2 = 0$. This concludes the proof.

Triangular inequality. Let's consider the distributions $\mu = \sum_{i=1}^{n} a_i \delta_{\boldsymbol{x}_i}$, $\nu = \sum_{i=1}^{n'} b_i \delta_{\boldsymbol{x}'_i}$, and a third distribution $\zeta = \sum_{i=1}^{w} c_i \delta_{\boldsymbol{z}_i} \in \mathcal{P}(\mathbb{R}^p)$ with strictly positive coefficient $(c_i)_{i=1}^{w}$. We have to show that $\mathcal{RPW}_2(\mu,\nu) \leq \mathcal{RPW}_2(\mu,\zeta) + \mathcal{RPW}_2(\zeta,\nu)$.

In order to ease the reading of the proof, we will make an abuse of notation. For any k $\in \{1,\dots,p\}$, we denote $\pi_{i,l}^{\boldsymbol{u}_k,*}$ (resp. $\pi_{l,j}^{\boldsymbol{u}_k,*}$) the coefficients of optimal transport plan between $\mu_{\boldsymbol{u}_k}$ and $\zeta_{\boldsymbol{u}_k}$ (resp. $\zeta_{\boldsymbol{u}_k}$ and $\nu_{\boldsymbol{u}_k}$). Since $\mu_{\boldsymbol{u}_k}$, $\nu_{\boldsymbol{u}_k}$ and $\zeta_{\boldsymbol{u}_k}$ are 1-d dimensional distributions, the composition of the optimal transport plans between $\mu_{\boldsymbol{u}_k}$ and $\zeta_{\boldsymbol{u}_k}$ and the transport plan between $\zeta_{\boldsymbol{u}_k}$ and $\nu_{\boldsymbol{u}_k}$ is an optimal transport plan between $\mu_{\boldsymbol{u}_k}$ and $\nu_{\boldsymbol{u}_k}$. This one is expressed as $\pi_{i,j}^{\boldsymbol{u}_k,*} = \sum_{l=1}^{w} \pi_{i,l}^{\boldsymbol{u}_k,*}\pi_{l,j}^{\boldsymbol{u}_k,*}/c_l$.

Therefore by definition of $\mathcal{RPW}_2$, we can write :

$$\begin{aligned}
\mathcal{RPW}_2(\mu,\nu) &\leq \left(\frac{1}{p}\sum_{k=1}^{p}\sum_{i,j=1}^{n,n'} \pi_{i,j}^{\boldsymbol{u}_k,*}\|\boldsymbol{x}_i - \boldsymbol{x}'_j\|_2^2\right)^{\frac{1}{2}} \\
&\leq \left(\frac{1}{p}\sum_{k=1}^{p}\sum_{i,j=1}^{n,n'}\sum_{l=1}^{w} \frac{\pi_{i,l}^{\boldsymbol{u}_k,*}\pi_{l,j}^{\boldsymbol{u}_k,*}}{c_l}\|\boldsymbol{x}_i - \boldsymbol{x}'_j\|_2^2\right)^{\frac{1}{2}} \\
&\leq \left(\frac{1}{p}\sum_{k=1}^{p}\sum_{i,j,l=1}^{n,n',w} \frac{\pi_{i,l}^{\boldsymbol{u}_k,*}\pi_{l,j}^{\boldsymbol{u}_k,*}}{c_l}\|(\boldsymbol{x}_i - \boldsymbol{z}_l) + (\boldsymbol{z}_l - \boldsymbol{x}'_j)\|_2^2\right)^{\frac{1}{2}}
\end{aligned} \tag{21}$$

Using the Minkowski inequality, we have:

$$\begin{aligned}\mathcal{RPW}_2(\mu,\nu) &\leq \left(\frac{1}{p}\sum_{k=1}^{p}\sum_{i,j,l=1}^{n,n',w}\frac{\pi_{i,l}^{\boldsymbol{u}_k,*}\pi_{l,j}^{\boldsymbol{u}_k,*}}{c_l}\|\boldsymbol{x}_i-\boldsymbol{z}_l\|_2^2\right)^{\frac{1}{2}}+\left(\frac{1}{p}\sum_{k=1}^{p}\sum_{i,j,l=1}^{n,n',w}\frac{\pi_{i,l}^{\boldsymbol{u}_k,*}\pi_{l,j}^{\boldsymbol{u}_k,*}}{c_l}\|\boldsymbol{z}_l-\boldsymbol{x}'_j\|_2^2\right)^{\frac{1}{2}}\\ &\leq \left(\frac{1}{p}\sum_{k=1}^{p}\sum_{i,l=1}^{n,w}\pi_{i,l}^{\boldsymbol{u}_k,*}\frac{\sum_{j=1}^{n'}\pi_{l,j}^{\boldsymbol{u}_k,*}}{c_l}\|\boldsymbol{x}_i-\boldsymbol{z}_l\|_2^2\right)^{\frac{1}{2}}+\left(\frac{1}{p}\sum_{k=1}^{p}\sum_{j,l=1}^{n',w}\frac{\sum_{i=1}^{n}\pi_{i,l}^{\boldsymbol{u}_k,*}}{c_l}\pi_{l,j}^{\boldsymbol{u}_k,*}\|\boldsymbol{z}_l-\boldsymbol{x}'_j\|_2^2\right)^{\frac{1}{2}}\end{aligned} \tag{22}$$

Since by definition $c_l=\sum_{j=1}^{n'}\pi_{l,j}^{\boldsymbol{u}_k,*}=\sum_{i=1}^{n}\pi_{i,l}^{\boldsymbol{u}_k,*}$, this leads to:

$$\begin{aligned}\mathcal{RPW}_2(\mu,\nu) &\leq \left(\frac{1}{p}\sum_{k=1}^{p}\sum_{i,l=1}^{n,w}\pi_{i,l}^{\boldsymbol{u}_k,*}\|\boldsymbol{x}_i-\boldsymbol{z}_l\|_2^2\right)^{\frac{1}{2}}+\left(\frac{1}{p}\sum_{k=1}^{p}\sum_{j,l=1}^{n',w}\pi_{l,j}^{\boldsymbol{u}_k,*}\|\boldsymbol{z}_l-\boldsymbol{x}'_j\|_2^2\right)^{\frac{1}{2}}\\ &\leq \mathcal{RPW}_2(\mu,\zeta)+\mathcal{RPW}_2(\zeta,\nu)\end{aligned}$$

This concludes the proof. Therefore $\mathcal{RPW}_2$ is a metric for the discrete distributions on $\mathbb{R}^p$.

A.1.2 Projection on canonical basis vector

Projecting onto the family of canonical vector basis is a choice that stems from a spanning constraint, and also a choice of simplicity. As we stated, for a given k, there may be several optimal transport plans between $\mu_{\boldsymbol{u}_k}$ and $\nu_{\boldsymbol{u}_k}$. This happens when, for a given k, there are bins where :

$$\boldsymbol{x}_i(k)=\boldsymbol{x}_j(k) \quad \text{and} \quad \boldsymbol{x}_i\neq\boldsymbol{x}_j \tag{23}$$

For continuous data (such as those that are the outputs of a GCN) this condition is particularly unlikely to happen. However, if we were projecting onto a non spanning vector family, that we call $\{v_u\}$, every bins lying in an orthogonal space of $\textbf{span}\{v_u\}$ would be projected on 0. This would lead on several couple of bins where the above condition (23) would be verified. In an extreme case where all bins of μ and ν would be in the orthogonal space, any transport plan would be optimal between projected distribution. Therefore, finding the OT plan which minimizes (12) would be equivalent to finding the optimal transport plan of Wasserstein distance. This is the reason why it is mandatory to find a family of vectors which spans R^p.

A natural choice is then to use the canonical basis from which it is easy to derive the identity of indiscernibles property (proven above) and from which the projection is costless on a numerical point of view. Anyway, in a different context from this work, where the distribution bins are not only continuous but also fit categorical data, some other spanning family may be more suitable. We hope that future works will take this idea to use specific family of vector to build specific distance (independently of the question of approximation of an existing OT distance) a step further.

A.2 Motivation and Interpretation of NCCML

We detail here some of the insights that led us to propose NCCML for ML.

Since we want to maintain a low complexity to train our model, a batch training is desirable. As a consequence and as said in section 4.3, the Large Margin Nearest Neighbor (LMNN) [15] loss was not appropriate because it works very locally and is not optimal with batch training. Indeed, LMNN tries to attract and repel points with elements of the datasets which are neighbours, according to their labels. On a batch training, this could lead to some *overfitting* where we try to attract points which should not be close even if they share the same label. This is even true the smaller the batch size we use.

An alternative is to use Neighborhood Component Analysis (NCA) [14] which provided a slightly better but limited performance. In reality, NCA is also a very locally method. Indeed it considers the probability $p(\mathcal{G}_i,\mathcal{G}_j)$ for two elements to have the same labels:

$$p_{\mathbf{\Theta}}(\mathcal{G}_i, \mathcal{G}_j) = \frac{\exp\left(-d_{\mathbf{\Theta}}^{\mathcal{RPW}_2}(\mathcal{G}_j, \mathcal{G}_i)^2\right)}{\sum_{k,k'} \exp\left(-d_{\mathbf{\Theta}}^{\mathcal{RPW}_2}(\mathcal{G}_k, \mathcal{G}_{k'})^2\right)} \tag{24}$$

Given this form of probability, it tries to maximize them for all elements which have effectively the same labels:

$$\max_{\mathbf{\Theta}} \sum_{\mathcal{G}_i \in \mathbb{G}} \sum_{\substack{\mathcal{G}_j \in \mathbb{G} \\ \mathcal{E}(\mathcal{G}_i)=\mathcal{E}(\mathcal{G}_j)}} p_{\mathbf{\Theta}}(\mathcal{G}_i, \mathcal{G}_j) \tag{25}$$

However, as one can see from Eq. (25), the probability of having the same labels is a softmax, so distant elements do not contribute a lot to these probability. It contains mostly local information. We believe that one could obtain better results by considering a more global criterion. Moreover, using a batch would be now advantageous since it will help the model to build good metric, even for k-NN (which requires a local fine metric) since the batch training will act as a regularization and will help to generalize.

An inspiration for that comes from NCMML [38] which proposes a loss function specifically built to increase performance of nearest mean classifier. This model also relies on a probabilistic model where the probability to belong on a class is given by a softmax which considers the distance to the mean of different classes. Obviously NCMML is not well suited for our tasks using kNN. Plus, it would require an additional layer of computation for computing barycenter with OT.

We took a compromise between NCA and the NCMML loss. The probability to be part of a class is given by a softmax which depends on the relative distance to different same label element (Eq. (11)). It has the advantage that the loss on a batch will be representative of the loss over the whole dataset, because the relative distance to different labels should remain the same also on subsamples of the dataset. Moreover it benefits from the batch training which acts as a regularizer. That finally leads to a better metric learned compared to NCA for k-NN as proven on our ablative study (Table. 3). Anyway, in a regular setting where we could use all datasets to build and train theses losses, NCCML would certainly shows worse results than LMNN and NCA.

The specific settings that is studied here, due to the requirement of scalability, forces to propose a loss different from the literature, that indeed brongs some improvement when compared to NCA.

A.3 Implementation details

Sequential implementation. A priori, it is necessary to compute all the transport costs between two distributions so as to calculate the optimal transport and this operation has a quadratic complexity. For most OT distance such as $\mathcal{W}_2$, since the complexity is dominated by the computation of the optimal transport plan, this was of no consequence. However for $\mathcal{RPW}_2$ (as well as for $\mathcal{SW}_2$) it becomes a critical aspect. Hopefully, there is no need to compute all the costs to find the optimal transport and the transport cost has no more than $n + m$ (given that the distributions have sizes n and m) non zero coefficients. This is why their complexity remains quasi-linear in $O(n \log n)$. The algorithm of the implementation referred to as "sequential implementation" in the core text can be found on Algorithm 2. The experiment on Section 5.2 assessed the quasi-linear complexity of this algorithm.

Quadratic implementation. In this second implementation, we compute all possible transport costs using a library of matrix multiplication, and then we multiply these costs by the optimal transport matrix. These operations allow us to benefit from the advantages of vectorization and to gain time compared to the sequential implementation, when n is not too large. This result is assessed experimentally in Section 5.2.

Both implementation can be found with this supplementary material.

Note: *In the reported experiments, we have seen that for $n < 1000$, it's better to use the quadratic implementation. Anyway this result strongly depends on the hardware used, and also on the dimension of the distribution support p. The scaling behavior of the two implementations is an interesting characteristic, showing than the proposed method can be implemented in a quasi-linear way. The*

Table 4: Graph datasets used in our experiments. #Graphs: number of graphs. #Nodes: average number of nodes. cont.: attributes have continuous values; lab.: attributes are labels. deg.: the featurattributes are degrees of nodes. q is the feature dimension.

Datasets	BZR	COX2	PROTEINS	ENZYMES	MUTAG	NCI1	IMDB-B	IMDB-M	CUNEIFORM
#Graphs	405	467	1113	600	188	4110	1000	1500	267
#Nodes	35.75	41.22	39.06	32.63	17.93	29.97	19.77	13	21.27
Node attributes	cont.	cont.	cont. / lab.	cont. / lab.	deg.	lab.	deg.	deg.	cont. / lab.
q	3	3	1 / 3	18 /3	4	38	135	88	3 / 3

Table 5: Typical runtimes in our experiments. The running time of WWL ($r = 2$) and $\mathcal{FGW}$ ($\alpha = 0.5$ except for IMDB datasets where it is set to 1) to calculate distances are also provided.

Datasets	BZR	COX2	PROTEINS [lab.]	ENZYMES [lab.]	MUTAG	NCI1	IMDB-B/M)	IMDB-M
Training time (s)	35	40	240	220	15	480	80/120	120
Distances comp. (s)	5	7	40	40	1	480	10/55	55
Dist. comp. (s) - WWL	16	25	200	30	2	1500	80/140	140
Dist. comp. (s) $\mathcal{FGW}$	240	270	1h	540	30	6h30min	1000/1400	1400

second comment is also that the method can be made rapid enough (and very competitive) with optimizations.

Algorithm 2 $\mathcal{RPW}_2$ - Sequential

Ensure: Build the distance between two discrete distributions μ and ν in $\mathcal{P}(\mathbb{R}^p)$.
Require: $\mu = \sum_{i=1}^{n} a_i \delta_{\boldsymbol{x}_i}$ and $\nu = \sum_{j=1}^{m} b_i \delta_{\boldsymbol{y}_j}$.
  Set $c = 0$.
  for each epoch $k \in \{1, \dots, p\}$ **do**
    Get σ_μ^k, σ_ν^k sort permutation of supports vectors k-th components.
    i.e $\boldsymbol{x}_{\sigma_\mu^k(0)}(k) \leq \cdots \leq \boldsymbol{x}_{\sigma_\mu^k(n-1)}(k)$ and $\boldsymbol{y}_{\sigma_\nu^k(0)}(k) \leq \cdots \leq \boldsymbol{y}_{\sigma_\nu^k(m-1)}(k)$.
    Set $T =$ *true*. Set $i, j = 0, 0$.
    Set $w_\mu, w_\nu = a_{\sigma_\mu^k(0)}, b_{\sigma_\nu^k(0)}$.
    while T == *True* **do**
      if $w_\mu < w_\nu$ **then**
        $c = c + w_\mu * ||\boldsymbol{x}_{\sigma_\mu^k(i)} - \boldsymbol{y}_{\sigma_\nu^k(j)}||_2^2$
        $i = i + 1$
        if $i == n$ **then**
          $T =$ *false*
        $w_\nu = w_\nu - w_\mu$
        $w_\mu = a_{\sigma_\mu^k(i)}$
      else
        $c = c + w_\nu * ||\boldsymbol{x}_{\sigma_\mu^k(i)} - \boldsymbol{y}_{\sigma_\nu^k(j)}||_2^2$
        $j = j + 1$
        if $j == m$ **then**
          $T =$ *false*
        $w_\mu = w_\mu - w_\nu$
        $w_\nu = b_{\sigma_\nu^k(j)}$
  return $\sqrt{\frac{c}{q}}$

A.4 Datasets

The characteristics of the datasets used are summarized in Table 4.

A.5 SGML - Datasets runtimes

The following Table 5 provides the typical runtimes for both training part and distance computation phases for the different datasets considered in this paper. We used the proposed quadratic implementation for all datasets. A `tensorflow` implementation is used during the training phase (to leverage the build-in functions for optimization and training) while the `numpy` implementation is used during the final distance computation. All running time experiments were conducted with a computer equipped with an Intel CORE i9900ks processor (62 GB of RAM) and GeForce RTX 3090 (24 GB of RAM).

The parameters are the same as in the experiments described in the paper. We fixed the depth of our GCN to $r = 4$.

As we can see despite lower theoretical complexity, the training time is bigger than distance computation. This is because the `numpy` implementation is much more efficient (especially in computing the sort operation) and these datasets are not large enough (in terms of the number of graphs) for `tensorflow` implementation catches up to `numpy` implementation. One clearly sees that the bigger the dataset (e.g., the NCI1 dataset), the lower the `numpy` implementation saves time.

A.6 $\mathcal{RPW}_2$ runtimes according to graph size

In section 5.2, we have not been able to extend the comparison of computation times between $\mathcal{RPW}_2$ and $\mathcal{SW}_2$ up to 10^8 size distributions in the same experimental conditions. The reason is that, beyond approximately 6 million points, we encountered a memory issue with $\mathcal{SW}_2$ on our Intel CORE i9900ks processor $\times$ 62 GB of RAM computer. It appears to be an implementation issue from POT toolboxes. Anyway, we have redone all the experiment with a computer with more RAM but a less powerful processor, an Intel Xeon Gold 5218 $\times$ 2 To of RAM. This amount of RAM is obviously overkill but it allows us to avoid any issue on the $\mathcal{SW}_2$ implementation. The results can be found in Figure 2. It confirms the calculated complexity on Section 4.4, asymptotically $\mathcal{RPW}_2$ scale better than $\mathcal{SW}_2$ since in our settings $p(= 5) < M(= 50)$.

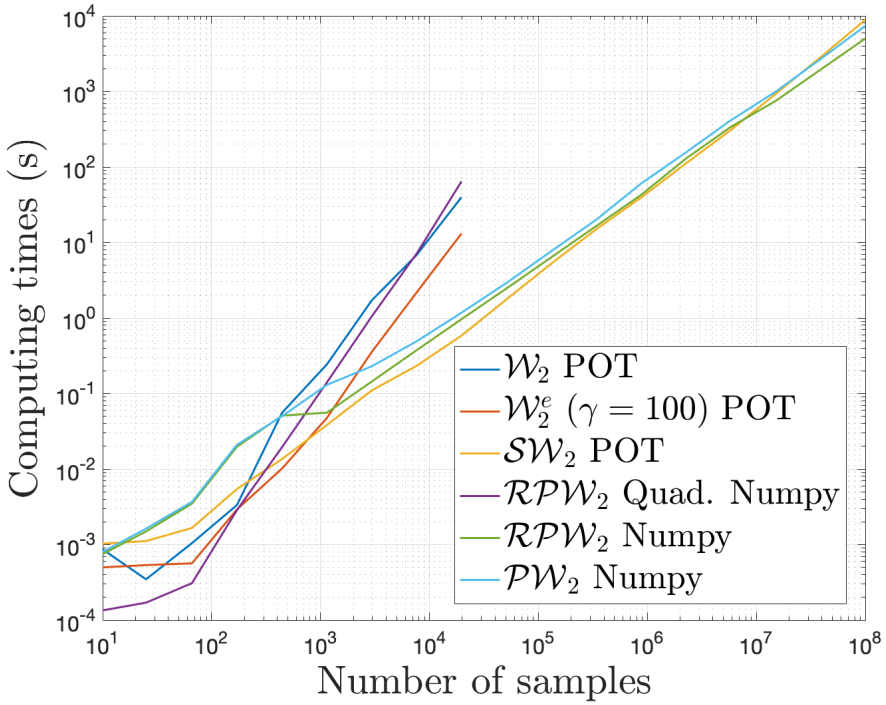

Figure 2: Run time comparisons 2.

Table 6: Ablative experiment with $\mathcal{FGW}$. Acc. is the accuracy. Δ is the difference in accuracy between the model of the column and the proposed one SGML whose results are on Table. 1. Red negative (resp. Green positive) number means that our model perform better (resp. worse). ✗ symbol means that we had infinite distance values with the default settings of FGW solver.

Dataset	$\mathcal{FGW}$	
Method	**Acc.**	**Δ**
BZR	81.70	- 3.91
COX2	78.51	- 1.28
MUTAG	83.16	- 6.84
NCI1	✗	✗
PROTEINS	✗	✗
IMDB-B	80.80	11.9
IMDB-M	✗	✗
ENZYMES	70.83	19.33

A.7 Additional details

ENZYMES. (discrete) The learning rate $0.999\ 10^{-2}$ was too heavy for NCA loss on ENZYMES, so we used $0.999\ 10^{-3}$ for this dataset. Accordingly we set the number of epochs to 20. However, we let the possibility to early stop at 10 epochs, meaning that the epochs number E becomes an hyper-parameter. $E = \{10, 20\}$.

PROTEINS. The above remark applies to PROTEINS (with continuous attributes). The learning rate was set to $0.999\ 10^{-4}$ and the epochs number E becomes an hyper-parameter $E = \{10, 20\}$.

CUNEIFORM. Since it has 30 different labels, the batch size has been set to 64.

ENZYMES. (continuous) It was trained in the same way as ENZYMES (discrete).

Extended vector attributes. We used a concatenation of continuous attributes and one-hot encoding of discrete attributes to build an extended vectors attributes. Since our method is a ML method it is pertinent to give all information we have and let the method to select the most relevant information. In case of PROTEINS, this choice was motivated because its node features are scalars which is not suitable for the adaptation procedure while in case of ENZYMES (continuous) and Cuneiform using only continuous attributes lead to poor results. This choice help to have more flexibility for SGCN to build the metric while avoiding to use more powerful but also more costly GCN.

A.8 FGW with k-NN

In the ablative study, we evaluated WWL with a k-NN to justify the design choice. Here, as a complement, we reproduced this experiment with $\mathcal{FGW}$. $\mathcal{FGW}$ has a parameter denoted $\alpha \in [0, 1]$ which sets the trade-off between the structure and the characteristics of the nodes in the distance computation. We performed a small grid search over this parameter $\alpha = [0.25, 0.5, 0.75]$. Except for IMDB datsaets where $\alpha = 1$ as in original paper. The results can be found in Table 6. One can see that the results are mitigated, FGW performs very well on some datasets and much less well on others. Moreover one could probably get even better results by doing a much larger hyperparameters tuning, as in the $\mathcal{FGW}$ original paper. Still, the present comparison is fair since, first, the grid search on the proposed method was also relatively small. Second, these results must be analyzed keeping in mind the significant difference in calculation time between the two methods (see Table 5). This illustrates also that doing a fine hyperparameter tuning with such expensive methods is not often feasible on very large data sets.

A.9 Limitations of this study

We discuss some of the limitations of the model and give some suggestions for improvements.

GCNs. To generate the distributions associated with the graphs, the model relies on a Graph Convolutionnal Neural network (GCN).

Because of this we can expect some sub-optimal behavior of the model in terms of expressiveness. Indeed, while they are very efficient to characterize graphs locally, GCNs tend to lose efficiency

when their depths increase. Although variations on their architectures have been proposed to solve this issue [39], it appears that most neural networks show similar results [6] and this defect seems to be intrinsic of their low-pass message passing scheme [40]. Therefore, new ways to efficiently characterize graphs at small and large scales could allow learning a better metric. In this regard, transformers are promising methods [41, 42]. Their ability to characterize context at different scales has already been successfully exploited in natural language processing tasks. Currently many attempts have been made in recent years to adapt them to graphs. However, these are difficult networks to train and their integration in SGML would not result in a simple and scalable metric learning model.

Performance. The model allows us to obtain an improvement in classification with k-NN as compared to the current methods. It is then more suitable for dealing with real datasets where new input are available after (or coming as graph streams) as the method do not need to be fully re-trained. However, performance with the k-NN remain inferior to those reached with a SVM. Thus in a critical real application (medical for example), where performance is of utmost importance, it is preferable to use the SVM. Additional work would be therefore necessary to gain more performance with the k-NN. This gain in performance could be acquired by introducing a different model of GCN so to generate the features, as mentioned above. But it could also done by making the model more complex. For example instead of considering uniform distributions from GCN features, we could introduce an attention mechanism that could modulate theirs weights on the distributions. This could give more flexibility to the model to build the metric, but at the cost of a more expensive training.

Theoretical. The work on the distance that we introduced here, $\mathcal{RPW}_2$, which is scalable and has a good behavior in our model, is currently methodological and driven by insight. As of today, we have not proven that it satisfies the triangular inequality so that it is not guaranteed that it is a true metric or not. This aspects remains to be clarified.

Opening to other tasks. Our work has been limited here to the k-NN for supervised classication. But other relevant classifiers with interesting properties where NCCML is not efficient enough could be consider, eg. the Nearest Class Mean [43]. Other tasks can also be considered such as clustering (k-means, ...) and regression (k-NN regression, ...). We believe that the present work is a first step on the goal of lowering the cost of many other tasks on graphs.