Neural Distance Embeddings
for Biological Sequences

NeurIPS 2021

Presenter: Joung Min Choi
Background

- **Edit distance** $D(s_1, s_2)$
  - The evolutionary distance between pairs of sequences
  - The minimum number of insertions, deletions or substitutions required to transform a string $s_1$ into another string $s_2$

- **Bioinformatics tasks**
  1) **Edit distance approximation**
     - The task of finding the distance or similarity between two strings
  2) **Hierarchical clustering**
     - Discovering the intrinsic hierarchical structure given by evolutionary history
     - Given a pairwise distance function, defining a tree with internal points corresponding to clusters and leaves to datapoints
3) Multiple sequence alignment (MSA)

- Aligning three or more sequences is used for the identification of active and binding sites
- Finding the global optimum of $N$ sequences: NP-complete
- Related task: Approximation of the Steiner string
  - Minimizing the sum of the distances (consensus error) to the sequences in a set

**Problem of the edit distance computation**

- Bound to a quadratic complexity and hardly parallelizable
- Computationally intractable in the accurate computation of similarities among multiple sequences
- Data independent classical algorithms
  - Cannot exploit the low-dimensional manifold assumption that characterizes the real world
Proposed approach

- **NeuroSEED (Neural Distance Embeddings)**
  - A general framework to produce representations for biological sequences
  - Learn an encoder function that preserves distances between the sequences and the vector space
    ⇒ Distance in the embedding space is correlated with the evolutionary distance $D$ between sequences

[Main components]
1) Embedding geometry
2) Encoder model
3) Decoder model
4) Loss function
1. **Embedding geometry**
   - Distance function $d$ between the embedded points
     : Defining the geometry of the embedding space
     
     - Mostly ignored by previous work, but critical to reflect the relations between the sequences

     \[
     \text{Manhattan} \quad d(p, q) = \|p - q\|_1 = \sum_{i=0}^{k} |p_i - q_i|
     \]
     \[
     \text{Euclidean} \quad d(p, q) = \|p - q\|_2 = \sqrt{\sum_{i=0}^{k} (p_i - q_i)^2}
     \]
     \[
     \text{square} \quad d(p, q) = \|p - q\|_2^2 = \sum_{i=0}^{k} (p_i - q_i)^2
     \]
     \[
     \text{cosine} \quad d(p, q) = 1 - \frac{p \cdot q}{\|p\| \|q\|} = 1 - \frac{\sum_{i=0}^{k} p_i q_i}{\sqrt{\sum_{i=0}^{k} p_i^2} \sqrt{\sum_{i=0}^{k} q_i^2}}
     \]
     \[
     \text{hyperbolic} \quad d(p, q) = \text{arccosh} \left(1 + \frac{\|p - q\|^2}{(1 - \|p\|^2)(1 - \|q\|^2)}\right)
     \]
Methods

2. **Encoder model**
   - Mapping sequences to points in the embedding space
   - Testing various models as encoder functions
     - Linear layer
     - MLP
     - CNN
     - GRU
     - Transformer with local and global attention
       - **Global**: every token queries all the others
       - **Local**: only queries its 2 neighbors
3. Loss function

- Minimize the mean squared error (MSE) between the sequences’ distance and its approximation as the distance between the embeddings

\[ L(\theta, S) = \sum_{s_1, s_2 \in S} (D(s_1, s_2) - \alpha d(\theta(s_1), \theta(s_2)))^2 \]

- MSE loss can be combined or substituted with other loss functions, depending on the application that the learned embeddings are used for:
  - Closest string retrieval: triplet loss
  - Hierarchical clustering: Dasgupta’s cost
  - Multiple sequence loss: sequence reconstruction loss
4. Decoder

- Training an autoencoder to map sequences to and from a continuous space preserving distances using only pairs of sequence at a time

- Loss function
  1) Edit distance approximation component
     : MSE between the real edit distance and a sequence reconstruction
  2) Sequence reconstruction loss
     : mean element-wise cross-entropy loss of the outputs with the real sequences

\[
L(\theta, \theta') = (1 - \alpha) L_{ED}(\theta) + \alpha L_{R}(\theta, \theta')
\]

where
\[
L_{ED}(\theta) = \left( n^{-1} \sum_{i=0}^{n-1} \left( H(s_1[i], g_{\theta'}(f_{\theta}(s_1))[i]) + H(s_2[i], g_{\theta'}(f_{\theta}(s_2))[i]) \right) \right)^2
\]

and
\[
L_{R}(\theta, \theta') = \frac{1}{2n} \sum_{i=0}^{n-1} \left( H(s_1[i], g_{\theta'}(f_{\theta}(s_1))[i]) + H(s_2[i], g_{\theta'}(f_{\theta}(s_2))[i]) \right)
\]
Methods

- Issue
  - Only learns the discrete subspace of points to which the generator maps some sequence from the domain
  - Need to decode points that are outside the subspace hoping to retrieve the string that maps to the point in the subspace closes to it

⇒ During training, adding Gaussian noise to the embedded point in the latent space before decoding it

\[
L(\theta, \theta') = (1 - \alpha) \underbrace{L_{ED}(\theta)}_{\text{edit distance}} + \alpha \underbrace{L_{R}(\theta, \theta')}_{\text{reconstruction}}
\]

where

\[
L_{ED}(\theta) = \left( n^{-1} \sum_{i=0}^{n-1} d(f_{\theta}(s_1[i]), f_{\theta}(s_2[i])) \right)^2
\]

and

\[
L_{R}(\theta, \theta', \epsilon) = \frac{1}{2n} \sum_{i=0}^{n-1} (H(s_1[i], g_{\theta'}(f_{\theta}(s_1 + \epsilon_1[i]))) + H(s_2[i], g_{\theta'}(f_{\theta}(s_2 + \epsilon_2[i])))
\]
Related work

- **Limitation in the previous works**
  1) The lack of analysis of the geometry of the embedding space
  2) The range of tasks is limited to edit distance approximation (EDA) and closest string retrieval (CSR)

 [Summary of the previous and the proposed NeuroSEED approaches]
Experiments

- **Baseline methods**
  1) K-mer
   - Construct a vector where each entry corresponds with the number of occurrences of a k-mer in the sequence
  2) FFP
   - Another alignment-free method which looks at the Jensen-Shannon divergence between distributions of k-mers

- **Three datasets with different portions of the 16S rRNA gene**

<table>
<thead>
<tr>
<th>Qitta</th>
<th>RT988</th>
<th>Greengenes full-length 16S rRNA DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than 6M sequences up to 15 2 bp covering V4 region</td>
<td>6.7k sequences of length up to 46 5 bp covering V3-V4 regions</td>
<td>More than 1M sequences between 1,111 to 2.368</td>
</tr>
</tbody>
</table>
**Results:** Edit distance approximation

- **Metric:** \(\text{%RMSE} = \frac{100}{n} \sqrt{L(\theta, S)} = \frac{100}{n} \sqrt{\sum_{s_1, s_2 \in S} (ED(s_1, s_2) - n d(f_\theta(s_1), f_\theta(s_2)))^2} \) (\(n:\) the maximum sequence length)

  - An approximate average error in the distance prediction as a percentage of the size of the sequences

- **Baseline Geometry**
  - Chosen by best average performance
  - K-mers: Cosine
  - FFP: Jensen-Shannon divergence
  - Neural models: Euclidean distance

![Figure 2: % RMSE test set results (4 runs)](image)
Results: Edit distance approximation

- Hyperbolic space
  - In CNN, the hyperbolic space provides a 22% average RMSE reduction against the best competing geometry for each model
  - It provides significantly more efficient embeddings with the model reaching the elbow at ‘32’ dimension

Figure 3: Qualitative comparison in the Qiita dataset between the best performing baseline (5-mer with cosine distance) and the CNN in the Euclidean and hyperbolic space. For every test set sequence pair, predicted vs real distances are plotted, the darkness represents the density of points. The CNN model follows much more tightly the red line of the oracle across the whole range of distances in the hyperbolic space.

Figure 4: Edit distance approximation % RMSE on Qiita dataset for a global transformer with different distance functions.
Results: Unsupervised heuristics

1. Hierarchical clustering (HC)
   - Test models with no further tuning from the previous exp on a dataset of 10k unseen sequences from the Qiita
   - No statistical difference in the quality of HC produced with ground truth distances
   - Significant difference in Dasgupta's cost between different architectures and geometries
   - Tree construction time reduced more than 30 minutes

* Dasgupta's cost: A global objective function that can be associated with the hierarchical clustering trees
Results: Unsupervised heuristics

2. Multiple sequence alignment

- Clustal (The most popular MSA heuristics)
  - Formed by a phylogenetic tree estimation phase that produces a guide tree then used by a progressing alignment phase
  - Hierarchical clustering: bottleneck ⇒ Can be accelerated by NeuroSEED embeddings

- Alignment scores obtained from the attention models are not significantly different from those with the ground truth
- Relatively large variance in the performance

![Table of results]

Figure 6: Percentage improvement (average of 3 runs) in the alignment cost (the lower the better) returned by Clustal when using the heuristics to generate the tree as opposed to its default setting using NJ on real distances.
1. Hierarchical clustering (HC)
   - Use Dasgupta’s cost as loss function to embed sequences in the hyperbolic space

   \[
   C_{\text{Dasgupta}}(T; w) = \sum_{ij} w_{ij} \cdot \text{leaves}(T[i \lor j])
   \]

   - Given a rooted binary tree \( T \), for two datapoints \( i \) and \( j \), \( w_{ij} \) be their pairwise similarity, \( i \lor j \) their lowest common ancestor in \( T \) and \( T[i \lor j] \) the subtree rooted at \( i \lor j \).
   - In this work, \( w_{ij} = 1 - d_{ij} \), \( d_{ij} \) is the normalized distance between sequence \( i \) and \( j \).

Figure 7: Average Dasgupta’s cost of the various approaches with respect to the number of pairwise distances used in the RT988 and Qiita datasets. The performances are reported as the percentage increase in cost compared to the one of the Average Linkage (best performing). Embedding refers to the baseline [20] while Linear to the relaxed NeuroSEED approach. The attached number represents the dimension of the hyperbolic space used.
Results: Supervised heuristics with tailored loss functions

2. Steiner string approach to multiple sequence alignment

- Use a decoder from the vector space to convert the Steiner string approximation problem
  1) Training the autoencoder to map sequences to and from a continuous space preserving distances
  2) In testing, the encoder embeds all the sequences in a set, the geometric median point is found, and the decoder is used to find the approximation of the Steiner string

![Diagram showing the Steiner string approach](image)
Results: Supervised heuristics with tailored loss functions

- **Metric:** The average consensus error (average distance to the strings in the set)
- **Baselines**
  - Random sequence in the set (Random)
  - The centre string of the set (Centre)
  - Two competitive greedy heuristics proposed by [50] and [51] (greedy-1 and greedy-2)

![Table Image]

Figure 9: Average consensus error for the baselines (left) and NeuroSEED models (right).
Contribution of this paper

- Introduce a general framework to map sequences in geometric vector spaces
- Show how the hyperbolic space can bring significant improvements to the data-dependent analysis of biological sequences
- Propose several heuristic approaches to classical bioinformatics problems
- Provide significant running time reduction against classical baselines
My thoughts

- Good approach to test different combinations of the neural network model and geometry function
- Somewhat close to the review paper, since this approach is not the first proposed one

- Based on the combination, each dataset shows quite a large difference
  - Need to perform multiple experiments
  - Require time to find the optimal one, which would be much more than just using the original distance
  - Hard to use this framework in the practical manner

- Utilizing hyperbolic space only shows better performance for some task
  - Especially on the edit distance approximation
  - Depends on the neural network model

- Little bit hard to follow the manuscript as the paper was not organized in smoothly