Hyperbolic Neural Networks

Octavian-Eugen Ganea*
Department of Computer Science
ETH Zürich
Zurich, Switzerland
octavian.ganea@inf.ethz.ch

Gary Bécigneul*
Department of Computer Science
ETH Zürich
Zurich, Switzerland
gary.becigneul@inf.ethz.ch

Thomas Hofmann
Department of Computer Science
ETH Zürich
Zurich, Switzerland
thomas.hofmann@inf.ethz.ch
Use **hyperbolic space** instead of **Euclidean space**
for embedding data with a latent **hierarchical** structure
The volume of a ball grows **exponentially** with its radius!

Use **hyperbolic space** instead of **Euclidean space** for embedding data with a latent **hierarchical** structure.
Use **hyperbolic space** instead of **Euclidean space** for embedding data with a latent **hierarchical** structure.

The volume of a ball grows **exponentially** with its radius!

Similarly as for a tree: the number of nodes grows **exponentially** with the tree depth!
Use **hyperbolic space** instead of **Euclidean space** for embedding data with a latent **hierarchical** structure.

**Hot topic in ML** since


Image source: http://prior.sigchi.org
Difficulties

- HOW TO USE HYPERBOLIC EMBEDDINGS IN DOWNSTREAM TASKS?
- HOW TO FEED HYPERBOLIC EMBEDDINGS TO NEURAL NETS?
Difficulties

- HOW TO USE HYPERBOLIC EMBEDDINGS IN DOWNSTREAM TASKS?
- HOW TO FEED HYPERBOLIC EMBEDDINGS TO NEURAL NETS?

- basic Euclidean operations not defined in the hyperbolic space! e.g. vector addition should follow hyperbolic "straight-lines", i.e. geodesics
Difficulties

- HOW TO USE HYPERBOLIC EMBEDDINGS IN DOWNSTREAM TASKS?
- HOW TO FEED HYPERBOLIC EMBEDDINGS TO NEURAL NETS?

- basic Euclidean operations **not defined** in the hyperbolic space! *e.g. vector addition should follow hyperbolic ”straight-lines”, i.e. geodesics*

- neural networks **should not ignore** the hyperbolic geometry (e.g. how to enforce hyperbolicity of an RNN’s hidden states?)
Poincaré Ball

\[ \mathbb{D}_c^n = \{ x \in \mathbb{R}^n, c \|x\|_2^2 < 1 \} \]
Poincaré Ball

\[ \mathbb{D}^n_c = \{ x \in \mathbb{R}^n, c \| x \|_2^2 < 1 \} \]
Poincaré Ball

\[ \mathbb{D}_c^n = \{ x \in \mathbb{R}^n, c \| x \|_2^2 < 1 \} \]

\[ d_c(x, y) = \frac{2}{\sqrt{c}} \tanh^{-1} \left( \sqrt{c} \| x \oplus_c y \| \right) \]
Use **Gyro-vector spaces** to generalize **basic operations** and **neural networks** from Euclidean to hyperbolic spaces:

- Gyro-vs: - analogue of Euclidean vector spaces
  - used in relativity theory (speeds of particles are hyperbolic)

- **Vector addition** $x + y \iff x \oplus_c y$

- **Scalar multiplication** $rx \iff r \otimes_c x$
  - Closed form distance $d_c(x, y) = (2/\sqrt{c}) \tanh^{-1}(\sqrt{c}\| - x \oplus_c y\|)$
  - Closed form geodesics: $\gamma_{x \to y}(t) := x \oplus_c (-x \oplus_c y) \otimes_c t$
Our contributions

1) We connect Gyro-vs and Riemannian hyperbolic geometry

• Closed form $\exp_x(v)$, $\log_x(y)$
• Closed form parallel transport (move across tangent spaces)
Our contributions

2) Hyperbolic Feed-forward Neural Networks

• Möbius version of \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) (e.g. pointwise non-linearity):

\[
f^\otimes_c : \mathbb{D}_c^n \rightarrow \mathbb{D}_c^m, \quad f^\otimes_c(x) := \exp_0^c(f(\log_0^c(x)))
\]
Our contributions

2) Hyperbolic Feed-forward Neural Networks

- Möbius version of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ (e.g. pointwise non-linearity):

  $$f^\otimes_c : \mathbb{D}_c^n \rightarrow \mathbb{D}_c^m, \quad f^\otimes_c(x) := \exp_0^c(f(\log_0^c(x)))$$

- Matrix - vector multiplication:

  $$M^\otimes_c(x) = (1/\sqrt{c}) \tanh \left( \frac{\|Mx\|}{\|x\|} \tanh^{-1}(\sqrt{c}\|x\|) \right) \frac{Mx}{\|Mx\|}$$

  - Properties: matrix associativity, scalar-matrix associativity, preserved rotations
Our contributions

3) Hyperbolic Softmax layer - Multiclass Logistic Regression

- Hyperbolic hyperplane:
  $$\tilde{H}^c_{a,p} = \{ x \in \mathbb{D}^n_c : \langle -p \oplus_c x, a \rangle = 0 \}.$$

- **Theorem**: closed form of $$d_c(x, \tilde{H}^c_{a,p})$$

- Final MLR formula (based on Lebanon and Lafferty, 2004):
  $$p(y = k | x) \propto \exp \left( \frac{\lambda^c_{p_k} \| a_k \|}{\sqrt{c}} \sinh^{-1} \left( \frac{2 \sqrt{c} \langle -p_k \oplus_c x, a_k \rangle}{(1 - c \| p_k \oplus_c x \|^2) \| a_k \|} \right) \right)$$
Our contributions

4) Hyperbolic Recurrent Networks, e.g. hGRU

\[
\text{hyp-GRU} \leftarrow \begin{cases} 
    r_t = \sigma \log_0^c(W^r \otimes_c h_{t-1} \oplus_c U^r \otimes_c x_t \oplus_c b^r) \\
    \tilde{h}_t = \varphi^c((W \text{diag}(r_t)) \otimes_c h_{t-1} \oplus_c U \otimes_c x_t \oplus b) \\
    h_t = h_{t-1} \oplus_c \text{diag}(z_t) \otimes_c (h_{t-1} \oplus_c \tilde{h}_t) 
\end{cases}
\]

- Hyperbolic hidden states

**Theorem:** update-gate mechanism derived from time-warping invariance principle (via gyro-derivative and gyro-chain-rule)
Property: All our models recover their Euclidean variants when curvature $c \to 0$. 
Riemannian Optimization

- Both Euclidean and hyperbolic parameters

- Riemannian SGD:
  \[ x \leftarrow \exp_x^c(-\eta \nabla_x^R \mathcal{L}), \quad x \in \mathbb{D}_c^n \]

- Riemannian gradient:
  \[ \nabla_x^R \mathcal{L} = \left(\frac{1}{\lambda_x^c}\right)^2 \nabla_x \mathcal{L}, \quad \text{conformal factor} \quad \lambda_x^c = \frac{2}{1 - c\|x\|^2} \]

Image source: stackexchange.com
Experiments
1) Textual Entailment tasks (semantic + syntactic).

<table>
<thead>
<tr>
<th>Test Accuracy</th>
<th>SNLI</th>
<th>PREFIX-10%</th>
<th>PREFIX-30%</th>
<th>PREFIX-50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Euclidean RNN</td>
<td>79.34 %</td>
<td>89.62 %</td>
<td>81.71 %</td>
<td>72.10 %</td>
</tr>
<tr>
<td>Hyp RNN+FFNN, Eucl MLR</td>
<td>79.18 %</td>
<td>96.36 %</td>
<td>87.83 %</td>
<td>76.50 %</td>
</tr>
<tr>
<td>Fully Hyperbolic RNN</td>
<td>78.21 %</td>
<td>96.91 %</td>
<td>87.25 %</td>
<td>62.94 %</td>
</tr>
<tr>
<td>Fully Euclidean GRU</td>
<td>81.52 %</td>
<td>95.96 %</td>
<td>86.47 %</td>
<td>75.04 %</td>
</tr>
<tr>
<td>Hyp GRU+FFNN, Eucl MLR</td>
<td>79.76 %</td>
<td>97.36 %</td>
<td>88.47 %</td>
<td>76.87 %</td>
</tr>
<tr>
<td>Fully Hyperbolic GRU</td>
<td>81.19 %</td>
<td>97.14 %</td>
<td>88.26 %</td>
<td>76.44 %</td>
</tr>
</tbody>
</table>

All word and sentence embeddings have dimension 5.
## Experiments

2) **MLR experiments.**

Test F1 classification scores (%) for 4 subtrees of the WordNet tree.

<table>
<thead>
<tr>
<th>WordNet Subtree</th>
<th>Model</th>
<th>$D = 2$</th>
<th>$D = 3$</th>
<th>$D = 5$</th>
<th>$D = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANIMAL.N.01 3218 / 798</td>
<td>HYP log$_0$</td>
<td>47.43 ± 1.07</td>
<td>91.92 ± 0.61</td>
<td>98.07 ± 0.55</td>
<td>99.26 ± 0.59</td>
</tr>
<tr>
<td></td>
<td>EUCL</td>
<td>41.69 ± 0.19</td>
<td>68.43 ± 3.90</td>
<td>95.59 ± 1.18</td>
<td>99.36 ± 0.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>38.89 ± 0.01</td>
<td>62.57 ± 0.61</td>
<td>89.21 ± 1.34</td>
<td>98.27 ± 0.70</td>
</tr>
<tr>
<td>GROUP.N.01 6649 / 1727</td>
<td>HYP log$_0$</td>
<td>81.72 ± 0.17</td>
<td>89.87 ± 2.73</td>
<td>87.89 ± 0.80</td>
<td>91.91 ± 3.07</td>
</tr>
<tr>
<td></td>
<td>EUCL</td>
<td>61.13 ± 0.42</td>
<td>63.56 ± 1.22</td>
<td>67.82 ± 0.81</td>
<td>91.38 ± 1.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60.75 ± 0.24</td>
<td>61.98 ± 0.57</td>
<td>67.92 ± 0.74</td>
<td>91.41 ± 0.18</td>
</tr>
<tr>
<td>WORKER.N.01 861 / 254</td>
<td>HYP log$_0$</td>
<td>12.68 ± 0.82</td>
<td>24.09 ± 1.49</td>
<td>55.46 ± 5.49</td>
<td>66.83 ± 11.38</td>
</tr>
<tr>
<td></td>
<td>EUCL</td>
<td>10.86 ± 0.01</td>
<td>22.39 ± 0.04</td>
<td>35.23 ± 3.16</td>
<td>47.29 ± 3.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.04 ± 0.06</td>
<td>22.57 ± 0.20</td>
<td>26.47 ± 0.78</td>
<td>36.66 ± 2.74</td>
</tr>
<tr>
<td>MAMMAL.N.01 953 / 228</td>
<td>HYP log$_0$</td>
<td>32.01 ± 17.14</td>
<td>87.54 ± 4.55</td>
<td>88.73 ± 3.22</td>
<td>91.37 ± 6.09</td>
</tr>
<tr>
<td></td>
<td>EUCL</td>
<td>15.58 ± 0.04</td>
<td>44.68 ± 1.87</td>
<td>59.35 ± 1.31</td>
<td>77.76 ± 5.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.10 ± 0.13</td>
<td>44.89 ± 1.18</td>
<td>52.51 ± 0.85</td>
<td>56.11 ± 2.21</td>
</tr>
</tbody>
</table>
Experiments

Hyperbolic (left) vs Direct Euclidean (right) binary MLR used to classify nodes as being part in the GROUP.N.01 subtree of the WordNet noun hierarchy solely based on their Poincaré embeddings.
THANK YOU!

Please visit our website: hyperbolicdeeplearning.com

HYPERBOLIC DEEP LEARNING
A nascent and promising field

Octavian Ganea
is currently looking for postdoctoral positions!
Matrix-vector multiplication

We define:

\[ M^{\otimes c}(x) = (1/\sqrt{c}) \tanh \left( \frac{\|Mx\|}{\|x\|} \tanh^{-1}(\sqrt{c}\|x\|) \right) \frac{Mx}{\|Mx\|}, \]

Nice properties:

- Matrix associativity: \( M \otimes (N \otimes x) = (MN) \otimes x \)
- Compatibility with scalar multiplication: \( M \otimes (r \otimes x) = (rM) \otimes x = r \otimes (M \otimes x) \)
- Directions are preserved: \( M \otimes x /\|M \otimes x\| = Mx /\|Mx\| \) for \( Mx \neq 0 \)
- Rotations are preserved: \( M \otimes x = Mx \) for \( M \in O_n(\mathbb{R}) \)
Matrix-vector multiplication

\[ M^\otimes_c(x) = \frac{1}{\sqrt{c}} \tanh \left( \frac{\|Mx\|}{\|x\|} \tanh^{-1}(\sqrt{c}\|x\|) \right) \frac{Mx}{\|Mx\|}, \]

When the curvature \( c \) goes to zero, it recovers the usual matrix multiplication!

\[ \lim_{c \to 0} M^\otimes_c(x) = Mx \]